

Univerzitet u Nišu

Elektronski fakultet Niš



Seminarski rad

Sistemi za upravljanje bazama podataka

Sigurnost MariaDB baze podataka

Mentor:

Prof. dr Aleksandar Stanimirović

Student:

Svetlana Mančić 1423

Niš, jun 2024. godine

Sadržaj

1. Uvod	3
2. Sigurnost baze podataka	4
2.1. Bezbednosne pretnje	4
2.2. Bezbednosne mere	5
2.2.1. Kontrola pristupa	6
2.2.2. Enkripcija i autentifikacija	10
3. Sigurnost MariaDB baze podataka	12
3.1. Autentifikacija korisnika kod MariaDB	14
3.2. Privilegije kod MariaDB	16
3.3. Kontrola pristupa kod MariaDB	17
3.4. Enkripcija podataka u tranzitu	24
3.5. Enkripcija podataka u mirovanju	30
3.5.1. Upravljanje ključevima za enkripciju	30
3.5.2. File Key Management	31
4. Zaključak	36
Literatura	37

1. Uvod

Bezbednosni aspekti u razvoju softvera i skladištenju podataka su ključni za zaštitu osetljivih informacija, održavanje poverenja korisnika i ublažavanje rizika povezanih sa zlonamernim napadima. Napad na kompaniju može biti dalekosežan, utičući na brojne aspekte poslovanja. Pored toga, zlonamerni napadi mogu dovesti do prekida rada, curenja podataka, krađe intelektualne svojine i drugih prekršaja, što vodi narušavanju reputacije i odnosa sa klijentima [1].

Kako pretnje napreduju u razvoju ogromnom brzinom, kompanije moraju da investiraju u snažne bezbednosne mere i sposobnost reagovanja na incidente. Davanje prioriteta bezbednosti je od suštinskog značaja za zaštitu informacija i korisnika i odbranu od zlonamernih napada. Primenom najboljih bezbednosnih praksi i principa u upravljanju podacima, organizacije izgrađuju i održavaju bezbedne i otporne sisteme, koji efikasno štite njihovu imovinu [1].

U drugom poglavlju biće reči o sigurnosti baza podataka, bezbednosnim pretnjama i bezbednosnim merama, koje se mogu preduzeti. Bezbednosne mere će biti navedene i ukratko opisane, radi boljeg razumevanja primera iz trećeg poglavlja.

U trećem poglavlju biće predložene bezbednosne mere na primeru MariaDB baze podataka, i demonstrirane na praktičnim primerima.

U četvrtom poglavlju biće dat zaključak.

2. Sigurnost baze podataka

U digitalnoj eri, kada je osnovna valuta informacija, baza podataka ima funkciju sefa, koji čuva neprocenljivo blago podataka. Ipak, usled bujice digitalnih transakcija i interakcija, digitalna skladišta suočavaju se sa stalno prisutnom pretnjom, koja ugrožava suštinu njihovog postojanja. Pretnja se ogleda u narušavanju poverljivosti (eng. *Confidentiality*), integriteta (eng. *Integrity*) i dostupnosti (eng. *Availability*) informacija, već dobro poznate trijade (*Confidentiality, Integrity, Availability - CIA*) u oblasti bezbednosti informacionih sistema. Poverljivost, integritet i dostupnost su osnovni bezbednosni zahtevi i od suštinskog su značaja za zaštitu podataka [2].

Poverljivost obezbeđuje da su osetljivi podaci, koji se skladište u bazi, dostupni samo ovlašćenim korisnicima. Uključuje sprečavanje neovlašćenog pristupa, otkrivanja ili curenja informacija. Integritet garantuje tačnost podataka u toku skladištenja i uključuje sprečavanje neovlašćenu modifikaciju podataka. Dostupnost obezbeđuje pristup skladištenim podacima od strane ovlašćenih korisnika i aplikacija. Uključuje sprečavanje zastoja, poremećaja ili *Denial of Service* (skraćeno *DoS*) napada, koji bi mogli uticati na dostupnost podataka [2].

2.1. Bezbednosne pretnje

Pretnje integritetu i poverljivosti podataka su višestruke, obuhvatajući širok spektar ranjivosti i vektora napada, koji koriste slabosti u strukturi bezbednosti baze podataka. Neke od najčešćih pretnji bezbednosti baze podataka uključuju [2]:

- Neovlašćeni pristup
- *SQL Injection* napadi
- *Denial of Service* napadi
- *Man-in-the-Middle* napadi
- Oštećenje podataka (eng. *data corruption*)
- Curenje podataka (eng. *data leaks*)

2.2. Bezbednosne mere

Da bi bezbednosni zahtevi bili ostvareni treba razviti jasnu i doslednu bezbednosnu politiku, koja će opisati bezbednosne mere, koje moraju biti sprovedene. Konkretno, treba odlučiti koji podaci će biti zaštićeni i koji korisnici mogu pristupati određenim delovima podataka. Dalje, bezbednosni mehanizmi DBMS-a i operativnog sistema koji je u osnovi, kao i eksterni mehanizmi, kao što su fizički pristup serverima, moraju biti iskorišćeni u sprovođenju politike. Treba naglasiti da se na više nivoa moraju preduzeti bezbednosne mere [1].

Neke od bezbednosnih mera podrazumevaju [2]:

- Zaštitne zidove (eng. *Firewall*), koji formiraju barijeru između servera baze podataka i spoljne mreže kontrolišući dolazni i odlazni mrežni saobraćaj na osnovu predefinisanih bezbednosnih pravila,
- Sisteme za detekciju i prevenciju napada (eng. *Intrusion Detection/Prevention Systems*), koji prate mrežni saobraćaj, detektuju sumnjivu ili zlonamernu aktivnost i obaveštavaju o potencijalnoj pretnji,
- Kontrolu pristupa (eng. *Access Control*), koja definiše ko može pristupiti resursima baze podataka i koje akcije mogu izvršavati korisnici,
- Autentifikaciju, koja predstavlja proveru identiteta korisnika, koji pokušavaju da pristupe bazi, i obezbeđuje da samo ovlašćeni korisnici pristupaju sistemu,
- Enkripciju podataka, što obezbeđuje zaštitu osetljivih podataka skladištenih u bazi, kao i zaštitu podataka u transportu,
- Maskiranje podataka, čime se štite osetljivi podaci i istovremeno održava funkcionalnost baze za ovlašćene korisnike i aplikacije.

Administrator ima bitnu ulogu u sprovođenju bezbednosnih aspekata dizajna baze podataka. Zajedno sa vlasnicima podataka, administrator učestvuje u razvoju bezbednosne politike. Ima poseban nalog, koji se naziva *system account*, i odgovoran je za celokupnu bezbednost sistema [3].

Administrator baze podataka zadužen je za [3]:

- Kreiranje novih korisničkih naloga - svakom novom korisniku ili grupi dodeljuje se *authorization ID* i lozinka.
- Pitanja obavezne kontrole pristupa - ako DBMS podržava obaveznu kontrolu pristupa, administrator mora dodeliti bezbednosne klase svakom objektu u bazi i bezbednosne dozvole svakom *authorization ID*-ju u skladu sa izabranom bezbednosnom politikom.
- Reviziju i praćenje baze podataka (eng. *Auditing and Monitoring*) - administrator održava *audit trail*, koji je u suštini log sa izmenama i *authorization ID*-jevima korisnika, koji su uneli izmene. Dodatno, administrator može održavati log svih akcija, uključujući čitanja od strane korisnika. Analizom istorije načina pristupa DBMS-u može se sprečiti kršenje bezbednosne politike i identifikovati sumnjivi obrasci pre nego što napadač uspe da upadne u bazu, ili može biti od značaja pri nalaženju napadača nakon detekcije upada.

2.2.1. Kontrola pristupa

Baza podataka jednog preduzeća skladišti ogromnu količinu informacija i obično ima više grupa korisnika. Većina korisnika ima ograničen pristup bazi podataka kako bi mogli da izvršavaju svoje zadatke. Nije poželjno da svi korisnici imaju neograničen pristup podacima, zato sistemi za upravljanje bazama podataka pružaju mehanizme kontrole pristupa podacima, koji dozvoljavaju sprovođenje izabrane bezbednosne politike preduzeća [3].

Kada je u pitanju kontrola pristupa, postoje tri glavna mehanizma na nivou DBMS-a [3]:

- Diskreciona kontrola pristupa (eng. *Discretionary access control - DAC*)
- Obavezna kontrola pristupa (eng. *Mandatory access control - MAC*)
- Kontrola pristupa bazirana na ulogama (eng. *Role-based access control - RBAC*)

Diskreciona kontrola pristupa bazira se na konceptima prava pristupa (tzv. privilegije) i mehanizmima dodeljivanja privilegija korisnicima. Privilegije omogućavaju korisniku pristup određenim objektima u bazi podataka (tabele i pogledi) na određeni način (na primer čitanje ili modifikaciju). Korisnik koji kreira objekat u bazi podataka automatski dobija sve privilegije nad tim objektom, uključujući i pravo da dodeljuje privilegije drugim korisnicima. DBMS potom prati kako se privilegije dodeljuju drugim korisnicima, ili oduzimaju, i obezbeđuje da u svakom trenutku

pristup objektima u bazi podataka bude omogućen samo korisnicima, koji imaju neophodne privilegije. SQL podrška diskrecionoj kontroli pristupa sprovedena je kroz komande GRANT i REVOKE [3].

GRANT komanda daje korisnicima privilegije nad tabelama i pogledima. Nekoliko privilegija može biti navedeno uključujući [3]:

- SELECT: pravo pristupa (čitanja) svih kolona tabele, uključujući i kasnije dodate kolone uz pomoć ALTER TABLE naredbe.
- INSERT (*column-name*): pravo dodavanja torki sa non-null ili null-default vrednostima u imenovanu kolonu tabele. Ako se navede samo INSERT, bez navođenja kolona, dodeljuju se privilegije za sve kolone, uključujući i one koje mogu biti kasnije dodate.
- DELETE: pravo brisanja torki iz tabele.
- REFERENCES (*column-name*): pravo definisanja stranog ključa u drugim tabelama, koji se odnosi na navedenu kolonu tabele. Ako se navede REFERENCES, bez naziva kolone, dodeljuje se pravo referenciranja svih kolona, uključujući i one koje mogu biti kasnije dodate.

Ako korisnik dobije privilegije sa GRANT OPTION, može prosledivati privilegije drugim korisnicima, sa ili bez GRANT OPTION [3].

Samo vlasnik može izvršavati CREATE, ALTER i DROP. Pravo izvršavanja navedenih komandi ne može se dodeliti ili oduzeti [3].

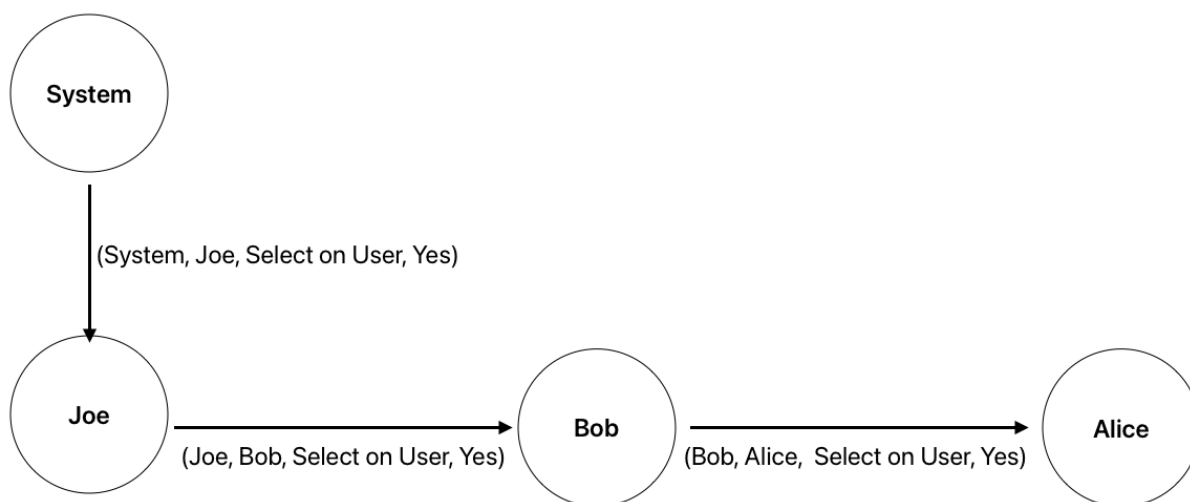
Nasuprot GRANT, REVOKE komanda služi za oduzimanje privilegija. Korisnik, koji je dodelio privilegije drugom korisniku, može oduzeti privilegije istom. Ako korisnik izvrši REVOKE komandu sa CASCADE ključnom rečju, oduzimaju se privilegije navedenom korisniku samo na osnovu GRANT komande koju je on prethodno izvršio. Ako su ti korisnici dobili prethodno dobili privilegije sa GRANT OPTION i prosledili ih drugim korisnicima, i privilegije tih korisnika se oduzimaju, osim ako nisu dobili privilegije kroz dodatne GRANT komande [3].

Kada se izvrši GRANT komanda, deskriptor privilegija (eng. *privilege descriptor*) se dodaje u tabelu deskriptora, koju održava DBMS. Deskriptor privilegije definiše ko je dodelio privilegije (*grantor*), kome su dodeljene privilegije (*grantee*), koje privilegije su dodeljene (*granted privilege*), uključujući i naziv objekta, i da li je GRANT OPTION uključen. Kada korisnik kreira tabelu ili

pogled i automatski dobije određene privilegije, dodaje se deskriptor privilegije, koji za *grantor* ima vrednost *system* [3].

Efekat niza GRANT komandi može se opisati grafom, gde čvorovi predstavljaju korisnike (*authorization ID-jeve*), a potezi ukazuju kako su privilegije dodeljene. Ako postoji poteg od korisnika1 do korisnika2 znači da je korisnik1 izvršio GRANT komandu i dodelio privilegije korisniku2. Poteg je označen deskriptorom privilegija [3].

Na slici 1 je prikazan graf dodele privilegija. Na početku Joe kreira tabelu User i dobija SELECT privilegiju od DBMS-a (poteg od System ka Joe). Zatim Joe izvršava GRANT komandu sa GRANT OPTION i dodeljuje SELECT privilegije za tabelu User Bobu (poteg od Joe ka Bob). Na kraju Bob izvršava GRANT komandu sa GRANT OPTION i dodeljuje SELECT privilegije za tabelu USER Alice (poteg od Bob ka Alice).



Slika 1. Graf dodele privilegija

Mehanizmi diskrecione kontrole pristupa, iako su generalno delotvorni, imaju određene slabosti. Naročito su podložni prevarama u vidu trojanaca, gde neovlašćeni korisnik može prevariti ovlašćenog korisnika da otkrije osetljive podatke. Mehanizmi **obavezne kontrole pristupe** treba da prevaziđu ovakve nedostatke diskrecione kontrole pristupa. Obavezna kontrola pristupa bazira se na bezbednosnoj politici celog sistema, koja ne može biti promenjena od strane individualnih korisnika [3].

Popularan model, koji se koristi kod obavezne kontrole pristupa, naziva se Bell-LaPadula model, koji je opisan objektima (tabele, pogledi, redovi, kolone), subjektima (korisnici, programi),

bezbednosnim klasama (eng. *security class*) i dozvolama (eng. *clearance*). Svaki objekat u bazi podataka ima dodeljenu bezbednosnu klasu, svaki subjekat ima dodeljenu dozvolu za bezbednosnu klasu, time su korisnicima nametnuta pravila čitanja i upisa za objekte u bazi podataka. DBMS odlučuje da li dati korisnik može da čita ili upisuje u dati objekat na osnovu određenih pravila, koja uključuju bezbednosni nivo dodeljen objektu i dozvole dodeljene korisniku. Ova pravila obezbeđuju da osetljivi podaci ne mogu biti prosleđeni korisniku bez neophodne dozvole [3].

Bezbednosne klase u sistemu su organizovane po delimičnom poretku, sa najbezbednijom klasom i najmanje bezbednom klasom. Pretpostavimo da postoje 4 klase: *top secret* (TS), *secret* (S), *confidential* (C) i *unclassified* (U). U sistemu, $TS > S > C > U$, gde $A > B$ znači da je klasa podataka A osetljivija od klase podataka B [3].

Bell-LaPadula model uvodi dva ograničenja za sve operacije čitanja i pisanja nad objektima baze podataka [3]:

- **Simple Security Property** - Subjektu S je dozvoljeno čitanje objekta O samo ako važi $class(S) \geq class(O)$. Na primer, korisnik sa TS dozvolom može da čita tabele sa C dozvolom, ali korisnik sa C dozvolom ne može da čita iz tabele, koja ima TS klasu.
- ***-Property** - Subjekt S može da piše u objekat O samo ako je $class(S) \leq class(O)$. Na primer, korisnik sa S dozvolom može da upisuje samo u objekte sa S ili TS klasama.

Ako je definisana diskreciona kontrola pristupa, ova pravila predstavljaju dodatna ograničenja. Stoga, da bi čitao ili pisao u objekte baze podataka, korisnik mora da ima potrebne privilegije, dodeljene kroz GRANT komande, dodatno, bezbednosne klase korisnika i objekta moraju zadovoljavati gore navedena ograničenja [3].

Kod diskrecione kontrole pristupa privilegije se dodeljuju na osnovu identiteta korisnika. Međutim, u stvarnosti se privilegije često povezuju sa poslom korisnika ili njegovom ulogom u preduzeću. Kod većine aktuelnih sistema, privilegije se dodeljuju ulogama (eng. *roles*). Uloga se definiše kao skup privilegija, koje ima korisnik za kog se vezuje ta uloga. Kada se pristupa sistemu, svaki korisnik navodi ulogu i ako mu je odobrena, on može iskoristiti privilegije dodeljene toj ulozi [4].

Politika kontrole pristupa definiše se u dva koraka [4]:

- Prvo, administrator definiše uloge i privilegije vezane za njih,

- Zatim, svakom korisniku se dodeljuje skup uloga, u kojima može da se nađe.

Uloge mogu biti hijerarhijski organizovane, kako bi se iskoristila propagacija privilegija duž hijerarhije [4].

Korisnik istovremeno može imati više uloga i više korisnika istovremeno može imati istu ulogu. Bitno je istaći da su uloge i grupe korisnika dva različita koncepta. Grupa korisnika je imenovan skup korisnika i drugih grupa, a uloga predstavlja imenovan skup privilegija i drugih uloga. Dok korisnik može aktivirati i deaktivirati ulogu po svom nahođenju, članstvo u grupi ne može se deaktivirati [4].

Uloge se mogu kreirati i brisati komandama CREATE ROLE i DROP ROLE. Uloge mogu biti dodeljene korisnicima ili drugim ulogama. Standardne GRANT i REVOKE komande mogu dodeljivati ili oduzimati privilegije ulogama ili *authorization ID-ijevima* [3].

Glavna prednost RBAC, uzimajući u obzir DAC i MAC, jeste što je pogodnija za komercijalna okruženja. U organizaciji nije bitan identitet osobe da bi ona pristupila sistemu, već njene odgovornosti. Takođe, kontrola pristupa bazirana na ulogama organizuje privilegije tako što mapira strukturu organizacije na uloge u hijerarhiji, koja se koristi za kontrolu pristupa [4].

2.2.2. Enkripcija i autentifikacija

Kada imamo podešenu kontrolu pristupa, treba razmotriti dodatne mehanizme bezbednosti, koje pruža DBMS, i preduzeti bezbednosne mere na više nivoa. Kada se bazi podataka pristupa sa bezbedne lokacije, možemo se osloniti na lozinku za autentifikaciju korisnika. Međutim, pristup bazi preko interneta donosi dodatne rizike, koji zahtevaju sofisticiraniji pristup od jednostavne autentifikacije lozinkom. U tom slučaju, neophodano je da korisnik potvrdi svoj identitet serveru, i obrnuto [3].

Kriptografija je nauka koja se bavi metodama očuvanja tajnosti informacija. Kriptografski algoritam predstavlja postupak, kojim se čitljivi podaci transformišu u nečitljive. Za postizanje bezbedne komunikacije, neophodan je jak algoritam, koji može biti javan i poznat napadaču, i tajni ključ, poznat samo izvoru i odredištu [5].

U odnosu na broj ključeva koji se koriste, kriptografski algoritmi mogu biti [5]:

- simetrični algoritmi - algoritmi kod kojih se isti ključ koristi za enkripciju i dekripciju podataka,
- asimetrični algoritmi - algoritmi kod kojih svaki korisnik ima par ključeva, javni i privatni.

Glavna slabost simetričnih algoritama je distribucija ključa ovlašćenim korisnicima. Ovaj problem simetričnih algoritama prevaziđen je kod asimetričnih algoritama korišćenjem para ključeva. Svaki korisnik ima javni i privatni ključ. Javni ključ je poznat svima i koristi se za enkripciju kada se podaci šalju korisniku, a privatni ključ je poznat samo korisniku i jedino se on može koristiti za dekripciju podataka. Kako se privatni ključ ne šalje, zagantovana je njegova tajnost [5].

Glavni problem asimetričnih algoritama jeste izbor ključeva. Asimetrični algoritmi se oslanjaju na postojanje *one-way* funkcija, čiju je inverznu funkciju veoma teško pronaći, pa je gotovo nemoguće zaključiti koji je privatni ključ, ako je poznat javni ključ i kriptografski algoritam [5].

DBMS koristi enkripciju za zaštitu informacija u situacijama kada ranije pomenuti bezbednosni mehanizmi nisu adekvatni. Na primer, napadač može ukrasti trake, koje sadrže podake, ili može prisluškivati kanale komunikacije. Skladištenjem i transportom enkriptovanih podataka, DBMS obezbeđuje da ukradeni podaci nisu od značaja napadaču. Otuda, možemo govoriti o [3]:

- Enkripciji podataka u mirovanju (eng. *Data-at-Rest Encryption*)
- Enkripciji podataka u tranzitu (eng. *Data-in-Transit Encryption*)

Tehnike enkripcije predstavljaju osnovu modernih načina autentifikacije. X.509 sertifikati predstavljaju digitalne dokumente, koji služe za potvrdu identiteta pojedinaca, organizacija ili uređaja na internetu. Sadrže informacije o identitetu nosioca sertifikata, kao što su ime, javni ključ, digitalni potpis i ime sertifikacionog organa (eng. *Certificate Authority - CA*), koji je izdao sertifikat. Javni ključ se koristi za enkripciju poruka, a digitalni potpis se koristi za proveru da li je poruku poslao nosilac privatnog ključa povezanog sa javnim ključem iz sertifikata. X.509 sertifikat je kao digitalna lična karta, koja omogućava enkriptovanu i autentifikovanu komunikaciju između dve strane [6].

Za dobijanje x.509 sertifikata najpre je neophodno generisati par ključeva, privatni i javni, uz pomoć nekog kriptografskog algoritma kao što je RSA ili ECC. Ovaj par ključeva će se koristiti za kreiranje i verifikaciju digitalnog potpisa, kao i enkripciju i dekripciju podataka. Kada je generisan par ključeva, kreira se zahtev za sertifikat, koji sadrži informacije o paru ključeva i identitetu nosioca sertifikata. Zahtev se šalje sertifikacionom organu od poverenja (to može biti javni CA kao što je Let's Encrypt ili neki privatni CA), koji potvrđuje identitet i izdaje validan x.509 sertifikat. Sertifikacioni organ verifikuje identitet podnosioca zahteva i potvrđuje zahtev za sertifikat obično slanjem email-a ili putem drugih oblika verifikacije identiteta. Kada CA potvrdi identitet i zahtev za izdavanje sertifikata, on izdaje x.509 sertifikat, koji sadrži javni ključ, informacije o identitetu i druge metapodatke, digitalno potpisane privatnim ključem sertifikacionog organa. Na kraju procesa, instalira se x.509 sertifikat na server, obično importovanjem fajla sertifikata u okviru konfiguracionog fajla [6].

3. Sigurnost MariaDB baze podataka

Ovo poglavlje bavi se glavnim bezbednosnim merama, koje se mogu primeniti u zaštiti MariaDB servera baze podataka uključujući i enkripciju podataka u tranzitu i u mirovanju. Primeri u seminarskom radu biće demonstrirani na MariaDB verziji 10.11.8, instaliranoj na Kali Linux-u. U trenutku pisanja seminarskog rada server baze podataka je bezbedan, jedino što je potrebno znati je kako ga konfigurisati.

Prvi korak u obezbeđivanju servera jeste ograničenje pristupa mašini na kojoj se on nalazi. Obično svaki korisnik ima svoje kredencijale za prijavljivanje i pripada nekoj unapred definisanoj grupi kada se prijavi. Kada se MariaDB server instalira na Linux-u, pokreće se od strane korisnika *mysql* u grupi *mysql*, tako da su sve kreirane datoteke sa podacima i logovima u vlasništvu *mysql:mysql*, a svim ostalim korisnicima ne bi trebalo dozvoliti ni pristup za čitanje. Datoteke sa podacima se podrazumevano nalaze na putanji */var/lib/mysql/*, a trenutne dozvole se mogu proveriti izvršenjem komande *ls -al* u terminalu (slika 2). Isti princip treba primeniti i kod datoteka sa ključevima i sertifikatima o kojima će biti više reči u poglavlju vezanom za enkripciju.

```
(kali㉿kali)-[/var/lib/mysql]
$ ls -al
total 123352
drwxr-xr-x  6 mysql mysql    4096 Jul 17 13:07 .
drwxr-xr-x 81 root  root    4096 Jun 27 18:28 ..
-rw-rw----  1 mysql mysql 417792 Jul 17 13:07 aria_log.00000001
-rw-rw----  1 mysql mysql   52 Jul 17 13:07 aria_log_control
drwx-----  2 mysql mysql    4096 Jun 28 06:27 db1
-rw-rw----  1 mysql mysql 12288 Jul 17 13:07 ddl_recovery-backup.log
-rw-rw----  1 mysql mysql   9 Jul 17 13:07 ddl_recovery.log
-rw-r--r--  1 root  root      0 Jun 27 18:28 debian-10.11.flag
-rw-rw----  1 mysql mysql   910 Jun 27 18:28 ib_buffer_pool
-rw-rw----  1 mysql mysql 12582912 Jun 27 18:28 ibdata1
-rw-rw----  1 mysql mysql 100663296 Jul 17 13:07 ib_logfile0
-rw-rw----  1 mysql mysql 12582912 Jul 17 13:07 ibtmp1
-rw-rw----  1 mysql mysql      0 Jun 27 18:28 multi-master.info
drwx-----  2 mysql mysql    4096 Jun 27 18:28 mysql
-rw-r--r--  1 root  root      15 Jun 27 18:28 mysql_upgrade_info
drwx-----  2 mysql mysql    4096 Jun 27 18:28 performance_schema
drwx-----  2 mysql mysql   12288 Jun 27 18:28 sys
```

Slika 2. Provera vlasništva i dozvola za datoteke

Nakon instalacije *mariadb-server* i *mariadb-client* paketa, dobro je izvršiti *mariadb-secure-installation* skriptu, koja korisnika vodi kroz dodatna bezbednosna podešavanja kao što su [7]:

- Podešavanje lozinke root naloga,
- Podešavanje autentifikacije preko unix socket-a,
- Onemogućavanje udaljenog root pristupa,
- Brisanje anonimnih korisnika,
- Brisanje test baze, kojoj mogu pristupiti anonimni korisnici.

Nakon instalacije servera, u konfiguraciji postoji parametar *bind-address* čija je vrednost 127.0.0.1, što znači da server sluša samo zahteve koji dolaze sa interfejsa localhost-a i ignoriše sve zahteve koji dolaze na ostale mrežne interfejse. Ovo je pogodno za testiranje na lokalnoj mašini, međutim, ako je potrebno da server baze podataka prihvata zahteve od spoljnog klijenta neophodna je konfiguracija ovog parametra. Na slici 3 je deo konfiguracionog fajla mariadb servera koji se odnosi na podešavanje adrese na kojoj sluša server.



```
bind-address = 192.168.64.4
```

Slika 3. Podešavanje parametra *bind-address*

Preporuka je konfiguracija firewall-a kako bi pristup serveru bio omogućen samo poznatim hostovima.

Neke od grešaka koje korisnici prave u konfiguraciji servera baze podataka uključuju ne podešavanje autentifikacije korisnika odmah, ne forsiranje enkriptovane komunikacije, davanje više privilegija nego što je neophodno korisnicima. U narednim poglavljima biće više reči o pomenutim problemima.

3.1. Autentifikacija korisnika kod MariaDB

Iako je ograničavanje pristupa serveru bitan faktor u zaštiti servera baze podataka, potrebno je uvesti dodatna ograničenja kako bi se sprečilo da kompromitovani hostovi pristupaju podacima. MariaDB pruža više mehanizama autentifikacije, koji zahtevaju identifikaciju korisnika pre izvođenja operacija nad podacima u bazi.

Pri kreiranju ili modifikovanju korisničkog naloga kroz GRANT, CREATE USER ili ALTER USER komande može se definisati *authentication plugin* za taj korisnički nalog navođenjem IDENTIFIED VIA klauzule, nakon čega se navodi *plugin* za autentifikaciju. Podrazumeva se da kada se kreira korisnički nalog bez definisanja *plugin-a*, koristi se *mysql_native_password* [8].

MariaDB pruža sledeće dodatke za *server-side* autentifikaciju [8]:

- ***mysql_native_password*** - podrazumevani dodatak, koji se koristi ako se pri kreiranju naloga ne navede neki drugi i *old_passwords* je setovan na 0. Koristi algoritam heširanja lozinke baziran na SHA-1. Upotreba ovog dodatka se ne preporučuje ukoliko se zahteva visok stepen bezbednosti lozinke. Ako neko može da prisluškuje protokol povezivanja i dođe do kopije mysql.user tabele, može iskoristiti ove informacije da se poveže sa MariaDB serverom. Za autentifikaciju ovim metodom na klijentskoj strani koristi se ***mysql_native_password*** dodatak. Na slici 4 je primer kreiranja korisnika kod kog se za autentifikaciju koristi *mysql_native_password*.

```

MariaDB [(none)]> set old_passwords=0;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> create user alex identified by 'password';
Query OK, 0 rows affected (0.004 sec)

```

Slika 4. *mysql_native_password* autentifikacija

- ***mysql_old_password*** - podrazumevani dodatak, koji se koristi ako se pri kreiranju korisničkog naloga ne navede drugi dodatak i *old_passwords* je setovan na 1. Kao i *mysql_native_password*, ne preporučuje se njegovo korišćenje, jer njegov algoritam heširanja nije više bezbedan, a dodatak je dostupan zbog kompatibilnosti sa prethodnim verzijama. Za autentifikaciju ovim metodom na klijentskoj strani koristi se ***mysql_old_password*** plugin.
- ***ed25519*** - baziran je na *Elliptic Curve Digital Signature* algoritmu, koji se koristi za bezbedno skladištenje i autentifikaciju korisnika. Za autentifikaciju ovim metodom na klijentskoj strani koristi se ***client_ed25519*** dodatak, koji lozinku hešira i potpisuje pre slanja serveru. Da bi bilo moguće korišćenje ovog *plugin*-a, u konfiguraciji se mora navesti *plugin_load_add = auth_ed25519*. Na slici 5 je primer kreiranja korisnika, kod kog se za autentifikaciju koristi *ed25519* plugin.

```

MariaDB [(none)]> create user aleks identified via ed25519 using password('password');
Query OK, 0 rows affected (0.008 sec)

```

Slika 5. *ed25519* autentifikacija

- ***gssapi*** (eng. *Generic Security Services Application Programming Interface*) - omogućava integraciju eksternih servisa za autentifikaciju, kao što je Kerberos. Ovaj metod povećava sigurnost omogućavanjem *single sign-on* mogućnosti i korišćenjem centralizovane infrastrukture za autentifikaciju, koji obezbeđuje Kerberos. Sa ovim *plugin*-om, na klijentskoj strani je kompatibilan ***auth_gssapi_client***.
- ***pam*** (eng. *Pluggable Authentication Module*) - framework za autentifikaciju kod Unix-like operativnih sistema. Omogućava implementaciju različitih scenarija autentifikacije, sa različitom kompleksnošću. Neki od pam modula, koji se mogu koristiti su: *pam_unix*, *pam_ldap*, *pam_ssh* i drugi. Kada je omogućena pam autentifikacija, MariaDB autentifikaciju delegira PAM framework-u, a od načina na koji je PAM konfigurisan zavisi kako će

autentifikacija biti izvršena. Koristi se samo kod Unix-like operativnih sistema. Na klijentskoj strani mogu se koristiti *dialog* i *mysql_clear_password* dodaci, koji su kompatibilni sa pam na serverskoj strani. Mana i jednog i drugog je što se lozinka šalje u *clear-text*-u, pa je neophodno da se pored autentifikacije obavezno podesi enkripcija u tranzitu, kako lozinku ne bi mogao da vidi naovlašćeni korisnik.

- ***unix_socket*** - omogućava korisniku da koristi kredencijale operativnog sistema za povezivanje sa MariaDB serverom preko lokalnog *Unix socket* fajla, koji se definiše preko socket systemske promenljive. Njegova bezbednost je u snazi pristupa Unix korisniku, a ne u složenosti i tajnosti lozinke. Kako bezbednost ne zavisi od lozinke, mogu se diskutovati njegove prednosti i slabosti. Prednost bi bila što nema lozinke koja se može *brute force*-ovati ili slučajno otkriti. Ovaj način autentifikacije nije pogodan kada više Unix korisnika pristupa jednom MariaDB korisničkom nalogu.
- ***named_pipe*** - koristi se samo kod Windows-a.

Od MariaDB 10.4 može se definisati više načina autentifikacije za jedan nalog.

3.2. Privilegije kod MariaDB

Nakon autentifikacije, korisnici biraju jednu od uloga, koje su im dodeljene od strane administratora, kroz koje imaju određene privilegije za izvršavanje komandi nad podacima.

Dodela privilegija korisnicima ili ulogama vrši se kroz GRANT komande. Da bi korisnik mogao da izvrši GRANT komandu za neki od objekata, mora da ima GRANT OPTION privilegiju, kao i privilegije koje želi da dodeli. Kod MariaDB, privilegije se mogu podešavati na više nivoa [9]:

- **Globalne privilegije** se dodeljuju korišćenjem *.* za nivo privilegija. Uključuju administratorske privilegije i upravljanje korisničkim nalogima, kao i privilegije za sve tabele, funkcije i procedure. Čuvaju se u mysql.global_priv tabeli. Na slici 6 je primer dodele globalnih privilegija.

```
MariaDB [(none)]> grant select on *.* to 'maria';  
Query OK, 0 rows affected (0.003 sec)
```

Slika 6. Dodela globalnih privilegija korisniku maria

- **Privilegije nad bazom** dodeljuju se korišćenjem *db_name.** za nivo privilegija. Uključuju privilegije za kreiranje tabela i funkcija, kao i privilegije za sve tabele, funkcije i procedure u navedenoj bazi. Čuvaju se u mysql.db tabeli. Na slici 7 je primer dodele privilegija nad bazom.

```
MariaDB [(none)]> grant select on db1.* to 'maria';
Query OK, 0 rows affected (0.002 sec)
```

Slika 7. Dodela privilegija nad bazom korisniku maria

- **Privilegije nad tabelom** dodeljuju se korišćenjem *db_name.db_table* šablona za nivo privilegija. Uključuju privilegije za čitanje i izmenu podataka. Određene privilegije nad tabelama mogu se dodeliti za individualne kolone. Na slici 8 je primer dodele privilegija nad tabelom.

```
MariaDB [db1]> grant select,insert,update on db1.user to 'maria';
Query OK, 0 rows affected (0.002 sec)
```

Slika 8. Dodela privilegija nad tabelom korisniku maria

- **Privilegije nad kolonama** dodeljuju se navođenjem tabele i liste kolona. Omogućavaju kontrolisanje kolona koje mogu biti pročitane i izmenjene od strane korisnika. Na slici 9 je primer dodele privilegija nad kolonama.

```
MariaDB [db1]> grant select(firstname,lastname,age) on db1.user to 'maria';
Query OK, 0 rows affected (0.003 sec)
```

Slika 9. Dodela privilegija nad kolonama tabele user korisniku maria

- **Privilegije nad funkcijama** dodeljuju se navođenjem *FUNCTION db_name.routine_name*.
- **Privilegije nad procedurama** dodeljuju se navošenjem *PROCEDURE db_name.routine_name*.

3.3. Kontrola pristupa kod MariaDB

Kada je u pitanju kontrola pristupa, kod MariaDB bazirana je na ulogama. Uloga povezuje niz privilegija i pomaže većim organizacijama u kojima bi određeni broj korisnika imao iste privilegije, dok je ranije jedini način promene privilegija za grupu korisnika bio da se promene privilegije za svakog korisnika pojedinačno. Dodatno, više spoljnih korisnika je moglo biti dodeljeno istom

korisničkom nalogu i ne bi bilo načina da se vidi koji korisnik je zapravo odgovoran za koju akciju [10].

Upravljanje ulogama je jednostavno. Može postojati veći broj korisnika, koji imaju istu ulogu, sa identičnim privilegijama. Promena privilegija za tu ulogu direktno se odražava na privilegije koje imaju korisnici kojima je ona dodeljena [10].

Uloge se kreiraju sa CREATE ROLE, a brišu sa DROP ROLE [10]. Na slici 10 je primer kreiranja uloga admin, developer i analyst.

```
MariaDB [(none)]> create role admin;  
Query OK, 0 rows affected (0.003 sec)  
  
MariaDB [(none)]> create role developer;  
Query OK, 0 rows affected (0.003 sec)  
  
MariaDB [(none)]> create role analyst;  
Query OK, 0 rows affected (0.003 sec)
```

Slika 10. Kreiranje uloga

Recimo da želimo da admin ima sve privilegije nad bazom db1, da developer ima privilegije upisa, modifikacije i čitanja za tabelu user u bazi db1, a analyst može samo da čita iz tabele user u bazi db1. Privilegije ulozi se dodeljuju kroz GRANT, a REVOKE se koristi za oduzimanje privilegija. Privilegije bismo dodelili kao na slici 11.

```
MariaDB [db1]> grant all privileges on db1.* to admin;  
Query OK, 0 rows affected (0.003 sec)  
  
MariaDB [db1]> grant select,insert,update on db1.user to developer;  
Query OK, 0 rows affected (0.002 sec)  
  
MariaDB [db1]> grant select on db1.user to analyst;  
Query OK, 0 rows affected (0.003 sec)
```

Slika 11. Dodela privilegija ulogama

Korisnicima se uloga dodeljuje takođe kroz GRANT komandu i oduzima kroz REVOKE [10]. Recimo da baza ima korisnike marija, marko i jovan, uloge bi im bile dodeljene kao na slici 12. U primeru marija ima ulogu admina, marko je developer, a jovan analyst.

```

MariaDB [db1]> grant admin to marija;
Query OK, 0 rows affected (0.006 sec)

MariaDB [db1]> grant developer to marko;
Query OK, 0 rows affected (0.002 sec)

MariaDB [db1]> grant analyst to jovan;
Query OK, 0 rows affected (0.002 sec)

```

Slika 12. Dodela uloga korisnicima

Dodeljene privilegije mogu se videti uz pomoć SHOW GRANTS FOR (slika 13).

```

MariaDB [db1]> show grants for developer;
+-----+
| Grants for developer |
+-----+
| GRANT USAGE ON *.* TO `developer` |
| GRANT SELECT, INSERT, UPDATE ON `db1`.`t` TO `developer` |
| GRANT SELECT, INSERT, UPDATE ON `db1`.`user` TO `developer` |
+-----+
3 rows in set (0.001 sec)

```

Slika 13. Dodeljene privilegije ulozi developer

Kada se korisnik poveže sa serverom, može dobiti privilegije povezane sa ulogom setovanjem uloge uz pomoć SET ROLE. CURRENT ROLE funkcija vraća trenutni set uloga za sesiju, ako su podešene (slika 14) [10].

```

MariaDB [db1]> set role developer;
Query OK, 0 rows affected (0.000 sec)

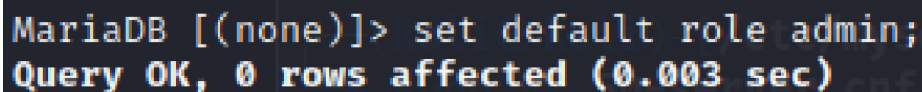
MariaDB [db1]> select current_role;
+-----+
| current_role |
+-----+
| developer    |
+-----+
1 row in set (0.001 sec)

```

Slika 14. Izbor uloge i prikaz trenutne uloge

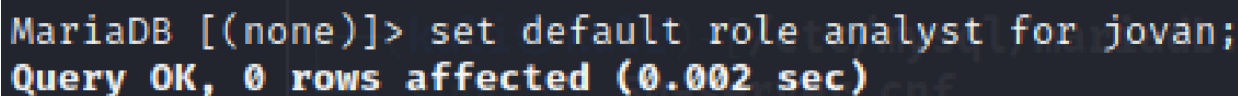
Samo uloge dodeljene direktno korisniku mogu biti podešene, uloge dodeljene drugim ulogama ne mogu [10].

Korisniku može biti setovana i podrazumevana uloga, koju korisnik automatski uzima pri konektovanju sa bazom. Da bi ovo bilo moguće, uloga koja se setuje mora prethodno da bude dodeljena korisniku kroz GRANT komandu. Za podešavanje podrazumevane uloge drugog korisnika, neophodne su *write* privilegije za *mysql* bazu. Na slikama 15 i 16 su primeri podešavanja podrazumevane uloge trenutnog korisnika i drugog korisnika. Podešavanje podrazumevane uloge za korisnika jovan izvršeno je sa root naloga, jer on ima potrebne privilegije.



```
MariaDB [(none)]> set default role admin;  
Query OK, 0 rows affected (0.003 sec)
```

Slika 15. Podešavanje podrazumevane uloge trenutnog korisnika



```
MariaDB [(none)]> set default role analyst for jovan;  
Query OK, 0 rows affected (0.002 sec)
```

Slika 16. Podešavanje podrazumevane uloge drugog korisnika

Zbog demonstracije kontrole pristupa, korisniku marija dodeljene su uloge admin, developer i analyst. U primeru na slici 17 može se videti da odmah nakon konektovanja izvršava se izbor baze, koji je odbijen jer korisnik nema podešenu podrazumevanu ulogu. Nakon uzimanja uloge analyst, moguć je pristup bazi db1. Potom je pokušao upis u tabelu user, koji takođe nije moguć, jer uloga analyst nema privilegiju upisa u tabelu user, već samo čitanja, što je demonstrirano na kraju SELECT naredbom.

```
(kali㉿kali)-[/var/lib/mysql]
$ mariadb -u marija -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 40
Server version: 10.11.8-MariaDB-1 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use db1;
ERROR 1044 (42000): Access denied for user 'marija'@ '%' to database 'db1'
MariaDB [(none)]> set role analyst;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> use db1;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [db1]> insert into user values ('Stefan',50);
ERROR 1142 (42000): INSERT command denied to user 'marija'@'localhost' for table `db1`.`user`
MariaDB [db1]> select * from user;
+-----+-----+
| name | age |
+-----+-----+
| Maria | 20 |
| Jovan | 25 |
| Marko | 35 |
+-----+-----+
3 rows in set (0.001 sec)
```

Slika 17. Demonstracija privilegija uloge analyst

U primeru na slici 18, korisnik marija bira ulogu developera. Upisuje vrednosti u tabelu user, ali nema privilegiju brisanja podatak iz iste. SELECT narednom može da pročita podatke iz tabele user. Takođe, kako uloga developera ima privilegije samo za tabelu user, a ne i za druge tabele u bazi db1, kada ima ulogu developera korisnik marija ne može da čita podatke iz drugih tabela u bazi db1.

```
MariaDB [db1]> set role developer;
Query OK, 0 rows affected (0.001 sec)

MariaDB [db1]> insert into user values ('Stefan',50);
Query OK, 1 row affected (0.002 sec)

MariaDB [db1]> delete from user where name='Jovan';
ERROR 1142 (42000): DELETE command denied to user 'marija'@'localhost' for table `db1`.`user`
MariaDB [db1]> select * from user;
+-----+-----+
| name | age |
+-----+-----+
| Maria | 20 |
| Jovan | 25 |
| Marko | 35 |
| Stefan | 50 |
+-----+-----+
4 rows in set (0.001 sec)

MariaDB [db1]> select * from cities;
ERROR 1142 (42000): SELECT command denied to user 'marija'@'localhost' for table `db1`.`cities`
```

Slika 18. Demonstracija privilegija uloge developer

Na kraju, u primeru na slici 19, korisnik marija podešava ulogu admina. Čita podatke iz tabele user, a potom briše torke gde je name='Jovan'. Zatim je opet prikazan sadržaj tabele user radi demonstracije brisanja podataka iz iste. Na kraju, kako admin ima sve privilegije nad svim tabelama u bazi db1, on može i da čita iz drugih tabela. SELECT naredbom prikazan je sadržaj tabele cities.

```
MariaDB [db1]> set role admin;
Query OK, 0 rows affected (0.001 sec)

MariaDB [db1]> select * from user;
+-----+-----+
| name | age |
+-----+-----+
| Maria | 20 |
| Jovan | 25 |
| Marko | 35 |
| Stefan | 50 |
+-----+-----+
4 rows in set (0.001 sec)

MariaDB [db1]> delete from user where name='Jovan';
Query OK, 1 row affected (0.002 sec)

MariaDB [db1]> select * from user;
+-----+-----+
| name | age |
+-----+-----+
| Maria | 20 |
| Marko | 35 |
| Stefan | 50 |
+-----+-----+
3 rows in set (0.001 sec)

MariaDB [db1]> select * from cities;
+-----+
| city |
+-----+
| Nis |
| Pirot |
| Beograd |
| Kragujevac |
| Leskovac |
+-----+
5 rows in set (0.001 sec)
```

Slika 19. Demonstracija privilegija uloge admin

3.4. Enkripcija podataka u tranzitu

Podrazumevano, MariaDB šalje podatke između servera i klijenta bez enkripcije. Ovo je generalno prihvatljivo kada su server i klijent na istom hostu ili na mrežama gde je bezbednost garantovana. Međutim, u slučajevima kada su server i klijent na odvojenim mrežama ili na mreži visokog rizika, nedostatak enkripcije predstavlja bezbednosni rizik, jer zlonamerni napadač može potencijalno da prisluškuje mrežni saobraćaj [11].

Na slici 20 prikazan je sadržaj mrežnih paketa snimljenih *tcpdump* alatom. Može se videti da je komunikacija između klijenta i servera neenkriptovana. Klijent je poslao upit za pribavljanje svih podataka iz tabele *user* (**`select * from user`**). Na slici se takođe vidi da tabela *user* ima kolone *name* i *age*, kao i vrednosti tih kolona. Ovo je jednostavan primer čiji je cilj bio da demonstrira kako bi napadač, koji prisluškuje mrežni saobraćaj, mogao da dođe do podataka skladištenih u bazi iako nema direktan pristup bazi i ne može da izvršava upite (*Man-In-The-Middle* napad), i važnost enkripcije podataka u tranzitu.

```
(kali@kali)-[/etc/mysql/mariadb.conf.d]
$ tshark -z follow,tcp,ascii,0 -P -r /home/kali/Documents/baze/mariadb.pcap
1   0.000000 192.168.64.4 → 192.168.64.4 MySQL 95 Request Query
2   0.000685 192.168.64.4 → 192.168.64.4 MySQL 251 Response TABULAR Respons
3   0.000724 192.168.64.4 → 192.168.64.4 TCP 72 35460 → 3306 [ACK] Seq=24 A

Follow: tcp,ascii
Filter: tcp.stream eq 0
Node 0: 192.168.64.4:35460
Node 1: 192.168.64.4:3306
23
.....select * from user
      179
.....*....def.db1.user.user.name.name .. !.....(....def.db1.user.user.age.
ovan.50.....Maria.20 ...
... " .
```

Slika 20. Neenkriptovan mrežni saobraćaj između servera i klijenta

Da bi se ovo izbeglo, MariaDB dozvoljava enkripciju podataka u tranzitu između servera i klijenta pomoću *Transport Layer Security (TLS)* protokola. Da bi MariaDB server koristio *TLS* mora da ima podršku za *TLS*. Ako nismo sigurni da li server ima podršku za *TLS* možemo to proveriti izvršavanjem upita kao na slici 21 [11].


```

MariaDB [db1]> show global variables like 'have_ssl';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| have_ssl      | DISABLED |
+-----+-----+
1 row in set (0.001 sec)

```

Slika 21. Provera TLS podrške

Sistemska promenljiva *'have_ssl'* može imati vrednost [11]:

- DISABLED, ako server ima podršku za TLS, ali nije omogućen,
- YES, ako server ima podršku za TLS i omogućen je,
- NO, ako server nema podršku za TLS.

Pored toga što server i klijent moraju da imaju podršku za *TLS*, neophodni su i *X.509* sertifikat, privatni ključ i lanac CA za verifikaciju *X.509* sertifikata servera. Ako se koristi *TLS* u oba smera, onda je isto potrebno i za klijenta. Mogu se koristiti i *self-signed* sertifikati kreirani uz pomoć *openssl* paketa [11].

Postoje 4 verzije TLS protokola [11]:

- TLSv1.0
- TLSv1.1
- TLSv1.2
- TLSv1.3

U nekim slučajevima ima smisla dozvoliti samo određene *TLS* verzije, na primer ako organizacija mora da ispuni specifične bezbednosne standarde ili ako je ranjivost pronađena u određenoj verziji *TLS* protokola i treba obezbediti da server ne koristi tu verziju protokola [11].

Na serverskoj strani, za podešavanje specifične verzije *TLS* protokola koristi se sistemski promenljiva *tls_version*, koja prihvata listu verzija odvojenih zapetama, i podešava se u okviru konfiguracionog fajla (slika 22). Verzija *TLS* protokola se može koristiti samo ako se nalazi u listi, sve druge verzije nisu dozvoljene [12].

```
[mariadb]
##Data-in-Transit Encryption
tls_version = TLSv1.2,TLSv1.3
```

Slika 22. Konfiguracija verzije *TLS* protokola

Na slici 23 je upit kojim se vrši provera verzija *TLS* protokola koje su dozvoljene u konfiguraciji.

```
MariaDB [db1]> show global variables like 'tls_version';
+-----+-----+
| Variable_name | Value          |
+-----+-----+
| tls_version   | TLSv1.2,TLSv1.3 |
+-----+-----+
1 row in set (0.002 sec)
```

Slika 23. Provera dozvoljenih verzija *TLS* protokola

Na klijentskoj strani, verzija *TLS* može se podestiti na isti način, u konfiguracionim fajlovima, ili navođenjem opcije `—tls-version` [12].

Kao što sam pomenula ranije, za konfiguraciju *TLS* protokola neophodno je generisati sertifikate i ključeve servera i klijenta. U primeru su *self-signed* sertifikati generisani uz pomoć *openssl* alata (slika 24). Najpre sam generisala ključ i sertifikat CA. Zatim sam generisala zahtev servera za izdavanje sertifikata i njegov ključ, a potom se na osnovu zahteva servera i CA sertifikata generiše i sertifikat servera. Postupak sam ponovila za generisanje sertifikata klijenta. Kompletan postupak napisan je u okviru *shell* skripte.

```

#CA key generating
openssl genrsa 4096 > /etc/mysql/ssl/ca-key.pem /etc/mysql/ssl/
#CA certificate generating
openssl req -new -x509 -nodes -days 365000 \
    -key /etc/mysql/ssl/ca-key.pem \
    -out /etc/mysql/ssl/ca-cert.pem
#Server key generating
openssl req -newkey rsa:2048 -days 365000 -nodes \
    -keyout /etc/mysql/ssl/server-key.pem \
    -out /etc/mysql/ssl/server-req.pem
openssl rsa -in /etc/mysql/ssl/server-key.pem \
    -out /etc/mysql/ssl/server-key.pem
#Signing server certificate
openssl x509 -req -in /etc/mysql/ssl/server-req.pem \
    -days 365000 -CA /etc/mysql/ssl/ca-cert.pem \
    -CAkey /etc/mysql/ssl/ca-key.pem \
    -set_serial 01 -out /etc/mysql/ssl/server-cert.pem
#Client key generating
openssl req -newkey rsa:2048 -days 365000 -nodes \
    -keyout /etc/mysql/ssl/client-key.pem \
    -out /etc/mysql/ssl/client-req.pem
openssl rsa -in /etc/mysql/ssl/client-key.pem \
    -out /etc/mysql/ssl/client-key.pem
#Signing client certificate
openssl x509 -req -in /etc/mysql/ssl/client-req.pem \
    -days 365000 -CA /etc/mysql/ssl/ca-cert.pem \
    -CAkey /etc/mysql/ssl/ca-key.pem \
    -set_serial 01 -out /etc/mysql/ssl/client-cert.pem
#Certificate verification
openssl verify -CAfile /etc/mysql/ssl/ca-cert.pem \
    /etc/mysql/ssl/server-cert.pem \
    /etc/mysql/ssl/client-cert.pem

```

Slika 24. Generisanje sertifikata

Poslednja korak rada sa sertifikatima je verifikacija sertifikata (slika 24, poslednja komanda), za šta sam iskoristila komandu *openssl verify*, čiji izlaz je prikazan na slici 25.

```

/etc/mysql/ssl/server-cert.pem: OK
/etc/mysql/ssl/client-cert.pem: OK

```

Slika 25. Verifikacija sertifikata

Kad imamo generisane sertifikate može se izvršiti konfiguracija vezana za *TLS* protokol u konfiguracionom fajlu MariaDB servera i MariaDB klijenta (*50-server.conf* i *50-client.conf* fajlovi u */etc/mysql/mariadb.conf.d* direktorijumu) [12].

Na slikama 26 i 27 prikazan je sadržaj konfiguracionih fajlova servera i klijenta vezan za konfiguraciju *TLS* protokola. Navodi se sertifikat CA, sertifikat i ključ servera, odnosno klijenta, i *TLS* verzije koje se mogu koristiti.

```
[mariadb]
#server-key.pem
server-key.pem
##Data-in-Transit Encryption
ssl-ca=/etc/mysql/ssl/ca-cert.pem
ssl-cert=/etc/mysql/ssl/server-cert.pem
ssl-key=/etc/mysql/ssl/server-key.pem
tls_version=TLSv1.2,TLSv1.3
```

Slika 26. Konfiguracija *TLS* protokola kod MariaDB servera

```
[client-mariadb]
#client-key.pem
client-key.pem
ssl-ca=/etc/mysql/ssl/ca-cert.pem
ssl-cert=/etc/mysql/ssl/client-cert.pem
ssl-key=/etc/mysql/ssl/client-key.pem
tls_version=TLSv1.2,TLSv1.3
```

Slika 27. Konfiguracija *TLS* protokola kod MariaDB klijenta

Potvrda da konekcija koristi *TLS* dobija se proverom *ssl_cipher* statusne promenljive (slika 28). Ako nije prazna, konekcija koristi *TLS* [12].

```
MariaDB [db1]> show session status like 'ssl_cipher';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| Ssl_cipher    | TLS_AES_256_GCM_SHA384 |
+-----+-----+
1 row in set (0.002 sec)
```

Slika 28. Provera da li konekcija koristi *TLS*

Nakon podešavanja enkripcije u tranzitu ponovila sam snimanje komunikacije između servera i klijenta. Na slici 29 se može videti da je komunikacija enkriptovana i čitanjem mrežnih paketa se ne može zaključiti upit koji je klijent poslao, kao ni sadržaj koji je dobio od servera.

```
(kali@kali)-[/etc/mysql]
$ tshark -z follow,tcp,ascii,0 -P -r /home/kali/Documents/baze/mariadb.pcap
1 0.000000 192.168.64.4 → 192.168.64.4 TCP 117 48276 → 3306 [PSH, ACK] Seq=1 Ack=1 Win=512 Len=45 TSval=772318935 TSecr=772304235 [TCP segment of a reassembled PDU]
2 0.000678 192.168.64.4 → 192.168.64.4 TCP 273 3306 → 48276 [PSH, ACK] Seq=1 Ack=46 Win=512 Len=201 TSval=772318936 TSecr=772318935 [TCP segment of a reassembled PDU]
3 0.000712 192.168.64.4 → 192.168.64.4 TCP 72 48276 → 3306 [ACK] Seq=46 Ack=202 Win=511 Len=0 TSval=772318936 TSecr=772318936

Follow: tcp,ascii
Filter: tcp.stream eq 0
Node 0: 192.168.64.4:48276
Node 1: 192.168.64.4:3306
45
....(.;.....J..D...?..(uR3.....T/\..TT....I..
201
.....i.....)..T....
..M.4....t.2M..i.L.....CV.z_..W..j`.....#..s=B0.....C).6.-..X.....P[.....T....zb;6.Y.bp.j..{A%...c....?.
Y....C.t...h...JT&|.....H.1z...c.:..XOD*....
```

Slika 29. Enkriptovan mrežni saobraćaj između servera i klijenta

Od MariaDB 10.5.2 verzije dostupna je *require_secure_transport* sistemska promenljiva. Kada je podešena, nebezbedne konekcije se automatski odbijaju. Pod bezbednim konekcijama podrazumeva se SSL/TLS, Unix socket ili named pipes. Takođe, za određene korisnike mogu se uvesti restrikcije vezane za TLS. Na primer, mogu se podesiti za korisnike koji pristupaju osetljivim podacima koji se šalju preko mreže. Za postavljanje restrikcija koriste se CREATE USER, ALTER USER ili GRANT izrazi.

Na slici 30 je primer naredbe kojom se podešava restrikcija da korisnik marko, koji pristupa sa hosta 192.168.64.4, mora da ima podešen ssl.

```
MariaDB [db1]> alter user 'marko'@'192.168.64.4' require ssl;
Query OK, 0 rows affected (0.002 sec)
```

Slika 30. Postavljanje restrikcije REQUIRE SSL

Na slici 31 je primer naredbe kojom se podešava restrikcija da korisnik marko, koji pristupa sa hosta 192.168.64.4, mora da koristi navedeni *ssl_cipher*.

```
MariaDB [(none)]> alter user 'marko'@'192.168.64.4' require cipher 'ECDH-RSA-AES256-SHA384';
Query OK, 0 rows affected (0.003 sec)
```

Slika 31. Postavljanje restrikcije REQUIRE CIPHER

U oba slučaja, konekcija je odbijena jer korisnik ne ispunjava podešena ograničenja (slika 32).

```
(kali@kali)-[~/Documents/baze]
$ mariadb -u marko -h 192.168.64.4 -p
Enter password:
ERROR 1045 (28000): Access denied for user 'marko'@'192.168.64.4' (using password: YES)
```

Slika 32. Odbijena konekcija

Korisnički nalog može imati različite definicije u zavisnosti od hosta sa kog se prijavljuje. Stoga, moguće je definisati različite *TLS* zahteve za isti korisnički nalog.

3.5. Enkripcija podataka u mirovanju

Enkripcija tabela onemogućava pristup podacima, čak iako dođe do krađe fizičkih uređaja na kojima se skladište. Od MariaDB verzije 10.1, moguća je enkripcija podataka u mirovanju (*Transparent Data Encryption - TDE*). Ovo podrazumeva skladištenje ključeva za enkripciju u okviru drugog sistema. Enkripcija je potpuno podržana za InnoDB skladište podataka. Podržana je i za Aria skladište, ali samo za tabele kreirane sa `ROW_FORMAT=PAGE`, koji je podrazumevani [13].

MariaDB dozvoljava fleksibilnu konfiguraciju podataka, koji se enkriptuju. Za InnoDB skladište moguće je [13]:

- enkripcija svih prostora tabela (eng. *tablespaces*),
- enkripcija određenih tabela,
- enkripcija svih prostora tabela osim određenih tabela.

Dodatno, preporučuje se i enkripcija log fajlova.

3.5.1. Upravljanje ključevima za enkripciju

Neophodna je upotreba dodatka za upravljanje ključevima i enkripciju (eng. *Key Management and Encryption Plugin*), koji je odgovoran za upravljanje ključevima, kao i za šifrovanje i dešifrovanje podataka. MariaDB podržava i korišćenje više ključeva za šifrovanje. Svaki ključ koristi 32 bitni integer za identifikator ključa. Ako dodatak podržava rotaciju ključeva, onda se i ključevi za šifrovanje mogu rotirati, što kreira novu verziju ključa za šifrovanje [13].

U zavisnosti od toga koji se *plugin* koristi, upravljanje ključevima se razlikuje. MariaDB trenutno nudi sledeće opcije [13]:

- *File Key Management Plugin* (dolazi uz MariaDB)
- *AWS Key Management Plugin* (zbog nekomatibilnosti licenci MariaDB izvornog koda i Amazon AWS C++ SDK može se preuzeti samo u vidu izvornog koda)
- *Hashicorp Key Management Plugin*

3.5.2. File Key Management

File Key Management, koji dolazi uz MariaDB, je najjednostavniji je za upravljanje ključevima i enkripciju i ima sledeće karakteristike [14]:

- Čita ključeve za šifrovanje iz *plain-text* datoteke
- Kao dodatni mehanizam zaštite, *plain-text* datoteka, koja sadrži ključeve, može se enkriptovati
- Podržava korišćenje više ključeva za šifrovanje
- Ne podržava rotaciju ključeva
- Podržava dva različita algoritma za šifrovanje podataka (*AES_CBC*, *AES_CTR*)

Kako bi enkripcija bila moguća, najpre se kreiraju ključevi za enkripciju. Format datoteke je na slici 33 [14].

```
<encryption_key_id1>;<hex-encoded_encryption_key1>  
<encryption_key_id2>;<hex-encoded_encryption_key2>
```

Slika 33. Format datoteke sa ključevima za enkripciju [14]

Datoteka za svaki ključ mora da sadrži sledeće informacije [14]:

- 32 bitni integer, koji predstavlja identifikator ključa, (na slici *encryption_key_id*)
- i ključ za enkripciju u hex-kodiranoj formi (na slici *hex-encoded_encryption_key*).

Za enkripciju se koristi *Advanced Encryption Standard* - *AES*, koji podržava 128b, 192b i 256b ključeve [14].

Za generisanje ključeva može se koristiti *openssl* paket. Identifikatori ključeva su 32b integer brojevi, koji ne moraju da se navode redom [14]. Na slici 34 je primer generisanja ključeva i identifikatora ključeva i upis u datoteku.

```
(kali㉿kali)-[~/Documents/baze]
$ cat generisanje-kljuceva.sh
(echo -n "1;" ; openssl rand -hex 32) | sudo tee -a /etc/mysql/encryption/keyfile
(echo -n "10;" ; openssl rand -hex 32) | sudo tee -a /etc/mysql/encryption/keyfile
```

Slika 34. Skripta za generisanje ključeva

Na slici 35 prikazan je sadržaj datoteke sa ključevima.

```
(kali㉿kali)-[~/Documents/baze]
$ cat /etc/mysql/encryption/keyfile
1;3cf6b683dfcab700b0a98d3382c7565b17ea4cda2d38c702f9b56760434e7083
10;e50571ee911e69b439cd5a416e1a8cde6a3a869e2ed28e6303234faf6a641da9
```

Slika 35. Datoteka sa ključevima za enkripciju

Identifikatori ključeva služe za referenciranje ključa za enkripciju. U prethodnom primeru mogu se referencirati ključevi sa identifikatorima 1 i 10 uz pomoć `ENCRYPTION_KEY_ID` opcije tabele ili sistemske promenljive `innodb_default_encryption_key_id`. Nije neophodno postojanje više ključeva [14].

Pošto se ključevi skladište u *plain-text* formatu na sistemu preporučuje se enkripcija datoteke sa ključevima kao dodatna bezbednosna mera. Pri enkripciji datoteke sa ključevima treba imati na umu da MariaDB podržava samo *AES Cipher Block Chaining - CBC* mod, dužine ključeva 128b, 192b ili 256b, ključ za enkripciju se kreira SHA-1 heširanjem uz pomoć lozinke maksimalne dužine 256 karaktera [14].

Lozinka za enkripciju se takođe može generisati uz pomoć *openssl* paketa. Za enkripciju datoteke sa ključevima može se koristiti *openssl enc* komanda, koja datoteku enkriptuje uz pomoć kreirane lozinke. Komanda *openssl enc* čita neenkriptovanu datoteku na putanji `/etc/mysql/encryption/keyfile.key` i kreira novu enkriptovanu datoteku na istoj putanji, `keyfile.enc`, koristeći lozinku sačuvanu u `keyfile.key`. Nakon enkripcije datoteke, originalna datoteka više nije potrebna i može se obrisati sa sistema [14]. Opisan postupak prikazan je na slici 36.


```
openssl rand -hex 128 | sudo tee /etc/mysql/encryption/keyfile.key

openssl enc -aes-256-cbc -md sha1 \
  -pass file:/etc/mysql/encryption/keyfile.key \
  -in /etc/mysql/encryption/keyfile \
  -out /etc/mysql/encryption/keyfile.enc
```

Slika 36. Enkripcija datoteke sa ključevima

Pošto datoteke ključeva sadrže osetljive podatke ne bi trebalo da iko može da ih pročita, pa je potrebno promeniti dozvole za pomenute datoteke.

Na slici 37 je deo konfiguracionog fajla servera vezan za enkripciju u mirovanju.

```
[mariadb]

##File Key Management
plugin_load_add = file_key_management
file_key_management_filename = /etc/mysql/encryption/keyfile.enc
file_key_management_filekey = FILE:/etc/mysql/encryption/keyfile.key
file_key_management_encryption_algorithm = aes_ctr

##InnoDB Encryption Setup
innodb_encrypt_tables = ON
innodb_encrypt_log = ON
innodb_encryption_threads = 4
innodb_default_encryption_key_id = 1

##Temp & Log Encryption
encrypt-tmp-disk-tables = 1
encrypt-tmp-files = 1
encrypt_binlog = ON
```

Slika 37. Konfiguracija enkripcije podataka u mirovanju

Iako je *plugin* uključen u MariaDB paket, nije instaliran od strane MariaDB po *default*-u. Da bi bio instaliran potrebno je navesti *—plugin-load* ili *—plugin-load-add* opciju kao argument *mysqld* ili navesti *plugin_load_add* opciju u konfiguracionom fajlu [14].

Ako je datoteka ključeva enkriptovana, pored *file_key_management_filename* sistemske promenljive, koja definiše putanju do datoteke ključeva, neophodno je navesti i *file_key_management_filekey* [14].

Sistemska promenljiva *file_key_management_filekey* može imati dve forme [14]:

- *Plain-text* lozinka za enkripciju (nije preporučeno jer se može videti u izlazu SHOW VARIABLES)
- Putanja do datoteke sa *plain-text* lozinkom (sa prefiksom FILE:)

Što se tiče algoritma za enkripciju, trenutno su podržani [14]:

- *AES_CBC* mod, koristi *Cipher Block Chaining*.
- *AES_CTR*, koristi dva različita moda u različitim kontekstima. Kada se enkriptuju stranice tabele, koristi se *Counter-CTR* mod. Kada se enkriptuju privremene datoteke (kada šifrirani tekst može biti veći od *plain-text-a*), koristi se *Galois/Counter Mode (GCM)*.

Za konfiguraciju algoritma služi *file_key_management_encryption_algorithm* sistemska promenljiva. Ovo ne treba mešati sa algoritmom koji MariaDB koristi za enkripciju datoteke ključeva, jer, kao što je već ranije rečeno, u tu svrhu podržan je samo *AES_CBC* [14].

Kada je konfigurisan plugin, može se dodati dodatna konfiguracija da bi se omogućila enkripcija InnoDB tabela [15]:

- *innodb_encrypt_tables*, omogućava automatsku enkripciju svih InnoDB prostora tabela, ali je dozvoljeno i kreiranje neenkriptovanih tabela (za forsiranje enkripcije koristi se vrednost *FORCE* ove sistemske varijable)
- *innodb_encrypt_logs*, omogućava enkripciju InnoDB redo log-a, kao i nekih privremenih fajlova kreiranih interno od strane InnoDB.
- *innodb_encryption_threads*, definiše broj pozadinskih za enkripciju.
- *innodb_default_encryption_key_id*, definiše podrazumevani ključ za enkripciju InnoDB prostora tabela.

Pored enkripcije prostora tabela, podešena je enkripcija privremenih fajlova i logova [14].

Na slici 38 su prikazane prve 5 linije datoteke, koja skladišti podatke tabele cities, u bazi db1. Tabela cities ima jednu kolonu (city varchar(50)) i kreirana je radi demonstracije enkripcije.

```
(kali@kali)-[/var/lib/mysql]
$ sudo strings ./db1/cities.ibd | head -n 5
Rdcj`e
jGK
|J      Z
EhRpO_K
LWC|
ERROR 1030 (HY000):
MariaDB [db1]>
ERROR 1030 (HY000):
MariaDB [db1]>
Query OK, 0 row affected
```

Slika 38. Enkriptovan sadržaj tabele cities

4. Zaključak

Cilj ovog rada bio je opis bezbednosnih mera i njihovog značaja za zaštitu osetljivih informacija, kao i demonstracija implementacije bezbednosnih mera na primeru MariaDB servera baze podataka. Na MariaDB serveru podešena je autentifikacija korisnika, definisane su i dodeljene uloge korisnicima u cilju demonstracije kontrole pristupa. Dodatno, podešena je i demonstrirana enkripcija u transportu i mirovanju.

Implementacijom robusnih kontrola pristupa, protokola za enkripciju i rigoroznih mehanizama monitoringa, organizacije mogu ublažiti rizike povezane sa neovlašćenim pristupom, krađom podataka i zlonamernim aktivnostima. Međutim, kako sajber pretnje neprestano evoluiraju, bezbednosna politika zahteva kontinuirano prilagođavanje i poboljšanje kako bi bila ispred potencijanih ranjivosti. Bezbednosne mere se moraju preduzeti na više nivoa kako bi se osigurala bezbednost podataka, od fizičke bezbednosti servera, preko operativnog sistema i servera baze podataka, do koda aplikacije.

Nakon uvoda, u drugom poglavlju diskutovano je o sigurnosti baza podataka, bezbednosnim pretnjama i bezbednosnim merama, koje se mogu preduzeti. Bezbednosne mere su navedene i ukratko opisane, radi boljeg razumevanja primera iz trećeg poglavlja.

U trećem poglavlji predložene su bezbednosne mere na primeru MariaDB baze podataka, i demonstrirane na praktičnim primerima.

Literatura

- [1] Importance of cyber security: Benefits and Disadvantages, <https://sprinto.com/blog/importance-of-cyber-security/> (pristup: 28.05.2024.)
- [2] Network Security Concepts, *CCNAv7: Enterprise Networking, Security and Automation*
- [3] R. Ramakrishnan, J. Gehrke, *Database Management Systems Third Edition*, McGraw-Hill Higher Education, 2003
- [4] M.Gertz, S. Jajodia, *Handbook of Database Security - Applications and Trends*, Springer, 2008
- [5] W. Stallings, *Cryptography and Network Security - Principles and Practice*, Pearson Education Inc, 2011
- [6] X.509 certificate: What is it and How it Works? <https://emudhra.com/blog/x509-certificate> (pristup: 06.06.2024.)
- [7] mariadb-secure-installation, <https://mariadb.com/kb/en/mariadb-secure-installation/> (pristup: 12.06.2024.)
- [8] Pluggable Authentication Overview, <https://mariadb.com/kb/en/pluggable-authentication-overview/> (pristup: 18.06.2024.)
- [9] GRANT, <https://mariadb.com/kb/en/grant/> (pristup: 25.05.2024.)
- [10] Roles Overview, https://mariadb.com/kb/en/roles__overview/ (pristup: 18.06.2024.)
- [11] Secure Connections Overview, <https://mariadb.com/kb/en/secure-connections-overview/> (pristup: 16.06.2024.)
- [12] Securing Connections for Client and Server, <https://mariadb.com/kb/en/securing-connections-for-client-and-server/> (pristup: 16.06.2024.)
- [13] Data-at-Rest Encryption Overview, <https://mariadb.com/kb/en/data-at-rest-encryption-overview/> (pristup: 15.06.2024.)
- [14] File Key Management Encryption Plugin, <https://mariadb.com/kb/en/file-key-management-encryption-plugin/> (pristup: 15.06.2024.)

[15] InnoDB Encryption Overview, <https://mariadb.com/kb/en/innodb-encryption-overview/>
(pristup: 15.06.2024.)