

**MySQL - SIGURNOST BAZA PODATAKA**

Seminarski rad

Studijski program: Računarstvo i Informatika

Modul: Bezbednost računarskih sistema

Predmet: Sistemi za upravljanje bazama podataka

Student:

Milan Stanković, br. ind. 1407

Niš, maj 2022. godina

Mentor:

Prof. Doc. Aleksandar Stanimirović

UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

Univerzitet u Nišu

Elektronski Fakultet

**MySQL – SIGURNOST BAZA PODATAKA**

Master akademske studije

**SEMINARSKI RAD:**

**Studijski program:** Računarstvo i Informatika

**Modul:** Bezbednost Računarskih Sistema

**Predmet:** Sistemi za upravljanje bazama podataka

**Student:** Milan Stanković, br. ind. 1407

**Emali:** milan.mixy.stankovic@elfak.rs

**Mentor:** Prof. Doc. Aleksandar Stanimirović

Niš, maj 2022. godina

**SADRŽAJ**

[**1.** **Uvod** 3](#_Toc104500552)

[**2.** **Bezbednost kod MySQL baze podataka** 4](#_Toc104500553)

[2.1. Generalni problemi sa bezbednošću 4](#_Toc104500554)

[2.1.1. Ukratko o rešenjima problema 5](#_Toc104500555)

[**3.** **Održavanje bezbednosti kod MySQL baze podataka** 6](#_Toc104500556)

[3.1. Formiranje i postavljanje bezbednih šifri 6](#_Toc104500557)

[3.1.1. Logovanje i prikaz lozinki 7](#_Toc104500558)

[3.1.2. Akcije sa administratorkse strane za bezbednost lozinki 8](#_Toc104500559)

[3.2. Sistem kontrole pristupa i privilegija kod MySQL-a 8](#_Toc104500560)

[3.2.1. Privilegije koje nudi MySQL 8](#_Toc104500561)

[3.2.2. Privilegije i GRANT tabele 10](#_Toc104500562)

[3.2.3. Specificiranje naloga i imena 10](#_Toc104500563)

[3.3. Kontrola pristupa 11](#_Toc104500564)

[3.3.1. Prva faza kontrole – verifikacija konekcije 12](#_Toc104500565)

[3.3.2. Druga faza kontrole – verifikacija zahteva 12](#_Toc104500566)

[3.3.3. Dodeljivanje lozinki naloga 12](#_Toc104500567)

[3.4. SQL Injection kod MySQL-a i prevencija 13](#_Toc104500568)

[3.5. Enkripcija podataka 14](#_Toc104500569)

[3.5.1. Postavljanje enkripcije 15](#_Toc104500570)

[3.5.2. Mogućnosti kod enkripcije podataka 16](#_Toc104500571)

[3.5.3. Kako funkcioniše enkripcija kod MySQL-a 16](#_Toc104500572)

[3.5.4. Primeri enkripcije 17](#_Toc104500573)

[3.6. Rezervne kopije i povratak podataka 18](#_Toc104500574)

[3.7. Sigurnost baza podataka sa strane infrastrukture 19](#_Toc104500575)

[3.7.1. DDoS napadi 19](#_Toc104500576)

[3.7.2. Replikacija MySQL baze podataka 21](#_Toc104500577)

[**4.** **Zaključak** 23](#_Toc104500578)

[**5.** **Literatura** 24](#_Toc104500579)

# **Uvod**

Bezbednost baze podataka obuhvata niz mera koje se koriste da bi se sistemi za upravljanje bazama podataka zaštitili od zlonamernih sajber-napada i nelegitimne upotrebe. Bezbednosni programi baze podataka su dizajnirani da zaštite ne samo podatke u bazi podataka, već i sam sistem za upravljanje podacima i svaku aplikaciju koja im pristupa od zloupotrebe, oštećenja i upada. Bezbednost obuhvata alate, procese i metodologije koje uspostavljaju bezbednost unutar okruženja baze podataka, a uz to podrazumeva dozvoljavanje ili onemogućavanje radnji korisnika na bazi podataka i objektima u njoj. Kada se kreira aplikacija baze podataka, bezbednosna politika je prvi korak. Politika bezbednosti aplikacije je lista bezbednosnih zahteva i pravila aplikacije koja regulišu pristup korisnika objektima baze podataka. Ovo poglavlje govori o aspektima bezbednosti aplikacija i karakteristikama MySQL baze podataka. [1]

MySQL je jedan od najpopularnijih sistema za upravljanje relacionim bazama podataka koji predstavlja najveću metu za napadače koji pokušavaju da se ušunjaju u baze podataka, i kako bi izvukli što više bitnih i osetljivih informacija iz sistema. Novoinstalirani MySQL server baze podataka može imati mnogo ranjivosti i rupa u zakonu. Pošto je bezbednost podataka od velike važnosti, obavezno je razumeti svaki aspekt MySQL bezbednosti. Ovaj aspekt posebno ima uogu u velikim poslovnim (*eng.* Enterprise) sistemima gde je svaka vrsta informacija ili podatka bitna, i ukoliko te informacije dospeju u pogrešne ruke, mogu imati veoma velike posledice po poslovanje firme. Pored toga ovo može znatno uticati na poverenje korisnika u sam sistem (što je dosta prisutno kod društvenih mreža) jer je veoma bitno da se ne naruši privatnost podataka krajnjih korisnika samog sistema.

Ovakvi sistemi kao što je MySQL pružaju mnogo mogućnosti za podešavanjem bezbednosti sistema baze podataka, međutim oni nisu automatski postavljeni već sam korisnik mora izvršiti analizu svojih potreba, a nakon toga vršiti postavljanje bezbednosnih podešavanja i ograničenja na bazom podataka.

U ovom radu glavni fokus će se posvetiti na reviziju i bezbednost široko upotrebljavane MySQL baze podataka. Pored toga biće stavljen akcenat na potencijalne mogućnosti podešavanje baze podataka odnosno DBMS-a kako bi se postigla maksimalna bezbednost za konkretan slučaj upotrebe (*eng.* Use Case) sistema.

# **Bezbednost kod MySQL baze podataka**

Kada se razmišlja o bezbednosti u okviru MySQL DBMS-a odnosno celokupnog njegovog sistema baze podataka, uvek bi se trebalo razmisliti o seširokom spektru mogućih karakteristika i funkcionalnosti i kako one utiču na bezbednost konkretnog MySQL servera i srodnih aplikacija. Kako je MySQL jedna od najpopularnijih relacionih baza podataka, velika je odgovornost na samom sistemu kako bi se administratorima i korisnicima omogućile sve potrebne funkcionalnosti kako bi se bezbednost podigla na što viši nivo.

**Opšti faktori koji utiču na bezbednost:**

To uključuje odabir dobrih lozinki, nedavanje nepotrebnih privilegija korisnicima, osiguranje bezbednosti aplikacija sprečavanjem SQL injekcija i oštećenja podataka i drugo. Sigurnost same instalacije. Datoteke sa podacima, datoteke evidencije i sve datoteke aplikacija vaše instalacije treba da budu zaštićene kako bi se osiguralo da neovlašćene strane ne mogu da ih čitaju ili pišu.

Kontrola pristupa i bezbednost unutar samog sistema baze podataka, uključujući korisnike i baze podataka kojima je odobren pristup bazama podataka, pogledima i uskladištenim programima koji se koriste u bazi podataka.

Mrežna bezbednost MySQL -a i samog sistema. Sigurnost je povezana sa dozvolama i mogućnostima za pojedinačne korisnike, ali može se takođe izvršiti i ograničavanje MySQL-a tako da bude dostupan samo lokalno na hostu MySQL servera ili na ograničenom skupu drugih hostova.

Potrebno je izvršiti blagovremeno kreiranje odgovarajuće rezervne kopije datoteka baze podataka, konfiguracije i datoteka evidencije. Takođe potrebno je sa sigurnošću imati rešenje za oporavak i neophodno je proveriti da li postoji mogućnost da se uspešno povrate informacije iz rezervnih kopija.

[2]

## **Generalni problemi sa bezbednošću**

Mnoge ranjivosti softvera, pogrešne konfiguracije ili obrasci zloupotrebe ili nepažnje mogu dovesti do kršenja. Nekoliko najpoznatijih uzroka i tipova sajber pretnji bezbednosti baze podataka su:

**Insajderska pretnja** – to je bezbednosni rizik iz jednog od sledeća tri izvora, a pritom svaki ima privilegovan pristup sistemu ili delu sistema: Zlonamerni insajder sa lošim namerama koji predstavlja osobu iz organizacije koja izlaže bazu podataka napadima. Anonimna osoba koja je na neki način, putem društvenog inženjeringa ili nekog drugog dobila pristup sistemu. Treći problem se javlja u velikim kompanijama kada se velikom broju zaposlenih daje pravo pristupa koji u nekom trenutku mogu zloupotrebiti te povoljnosti.

**Ljudski faktor / greška** – Slabe lozinke, deljenje lozinki, slučajno brisanje ili oštećenje podataka i druga nepoželjna ponašanja korisnika i dalje su uzrok skoro polovine prijavljenih narušavanja podataka.

**Iskorišćavanje ranjivosti softvera baze podataka** – Napadači stalno pokušavaju da izoluju i ciljaju ranjivosti u softveru, a softver za upravljanje bazom podataka je veoma vredna meta. Nove ranjivosti se otkrivaju svakodnevno, a sve platforme za upravljanje bazama podataka otvorenog koda i komercijalni proizvođači softvera za baze podataka redovno izdaju sigurnosne zakrpe. Međutim, ako ove zakrpe ne koristite brzo, baza podataka može biti izložena napadu. Čak i ako se primene zakrpe na vreme, uvek postoji rizik od napada nultog dana, kada napadači otkriju ranjivost, ali ona još uvek nije otkrivena i zakrpljena od strane dobavljača baze podataka.

**SQL injekcioni (*eng.* injection) napadi** – Pretnja specifična za bazu podataka uključuje upotrebu proizvoljnih SQL stringova napada u upitima baze podataka. Obično su to upiti kreirani kao proširenje obrazaca za web aplikacije ili primljeni putem HTTP zahteva. Svaki sistem baze podataka je ranjiv na ove napade, ako se programeri ne pridržavaju praksi bezbednog kodiranja i ako organizacija ne sprovodi redovno testiranje ranjivosti.

**Napadi prekoračenja bafera –** Prelivanje bafera se dešava kada proces pokušava da upiše veliku količinu podataka u blok memorije fiksne dužine, više nego što je dozvoljeno da zadrži. Napadači mogu koristiti višak podataka, koji se čuvaju na susednim memorijskim adresama, kao početnu tačku za pokretanje napada.

**Napadi uskraćivanja usluge (DoS/DDoS).** – U napadu uskraćivanja usluge (DoS), sajber kriminalac nadvladava ciljnu uslugu – u ovom slučaju server baze podataka, koristeći veliku količinu lažnih zahteva. Rezultat je da server ne može da izvrši prave zahteve stvarnih korisnika i često se ruši ili postaje nestabilan. U distribuiranom napadu uskraćivanja usluge (DDoS), lažni saobraćaj generiše veliki broj računara, koji učestvuju u botnetu koji kontroliše napadač. Ovo generiše veoma velike količine saobraćaja, koje je teško zaustaviti bez visoko skalabilne odbrambene arhitekture. Usluge zaštite od DDoS-a zasnovane na oblaku mogu se dinamički povećati kako bi se pozabavile veoma velikim DDoS napadima.

**Zlonamernih programa** – Malver je softver napisan da iskoristi ranjivosti ili da nanese štetu bazi podataka. Zlonamerni softver može da stigne preko bilo kog krajnjeg uređaja povezanog na mrežu baze podataka. Zaštita od zlonamernog softvera je važna na bilo kojoj krajnjoj tački, a posebno na serverima baza podataka, zbog njihove visoke vrednosti i osetljivosti.

### **Ukratko o rešenjima problema**

Prethodno navedeni generalni problemi u poglavlju 2.1 predstavljaju uopštene (osnovne) probleme sa kojima se susreću organizacije koje koriste rešenja ovakvih vrsta baza podataka u našem konkretnom primeru MySQL baza podataka. U nastavku rada biće opisani detaljni slučajevi i rešenja kao i primeri koji prikazuju širu sliku o potencijalnom načinu rešavanja problema, ali će ovde biti ukratko navedena potencijalna rešenja.

* Osim MySQL root naloga, nikome ne treba dozvoljavati pristup korisničkoj tabeli u MySQL bazi podataka.
* Koristite naredbe **GRANT** i **REVOKE** za kontrolu pristupa MySQL-u.
* Lozinke nikad ne čuvati kao običan tekst, već ih čuvati kao heš vrednosti funkcija SHA2(), SHA1(), MD5() ili druge funkcije heširanja.
* Korišćenje zaštitne barijere (*eng.* Firewall) kod pristupnih tačaka bazi.
* Iako je mrežni port 3306 podrazumevani port za MySQL, trebalo bi da taj port bude zatvoren.
* Za prenos podataka uvek koristiti SSL (*eng.* Secure Socket Layer).

# **Održavanje bezbednosti kod MySQL baze podataka**

U ovom poglavlju biće detaljno obrađene sve mogućnosti i karakteristike koje nudi MySQL. Pored prikazanih funkcionalnosti biće opisan detaljan način upotrebe, kako ih upotrebiti.

**Namepena:** svi primeri konzolnih komandi su prikazane za primer Unix operativnih sistema!

## **Formiranje i postavljanje bezbednih šifri**

Lozinke se javljaju u nekoliko konteksta unutar MySQL-a. Biće navedene smernice koje omogućavaju krajnjim korisnicima i administratorima da čuvaju ove lozinke bezbednim i izbegnu njihovo otkrivanje. Pored toga, dodatak **validate\_password** se može koristiti za sprovođenje politike prihvatljive lozinke o kom će biti više reči.

Kada se pokrene klijentski program da bi se povezao sa MySQL serverom, nije preporučljivo da se navede lozinka na način koji je izlaže otkrivanju drugih korisnika. Ovde su navedene metode koje se mogu koristiti da kako bi naveli svoju lozinku kada se pokreću klijentski programi, zajedno sa procenom rizika svake metode. Ukratko, najsigurnije metode su da klijentski program zatraži lozinku ili da je korisnik navede u ispravno zaštićenoj opcionoj datoteci.

Praksa je koristiti uslužni program (*eng.* Utility) **mysql\_config\_editor**, koji omogućava da skladištimo akreditive koji su nam neophodni za autentifikaciju u enkriptovanoj datoteci za prijavu pod nazivom **.mylogin.cnf** . Ovu datoteku mogu kasnije koristiti MySQL klijentski programi kako bi dobili akreditive za autentifikaciju za povezivanje sa MySQL serverom.

Koristimo –password=**password** ili -p**password**  kao što je dato na sledećem primeru:

**$> mysql -u testuser -ppassexample db\_name**

Naredba 1.

U ovom primeru je za korisnika **testuser** postavljena lozinka **passexample**. Ako je operativno okruženje podešeno da prikazuje trenutnu komandu u naslovnoj traci prozora terminala, lozinka ostaje vidljiva sve dok je komanda pokrenuta, čak i ako je komanda izašla iz vida u oblasti sadržaja prozora. Postoji i interaktivni način unošenja lozinke korišćenjem sledeće naredbe:

**$> mysql -u francis -p db\_name**

**Enter password: \*\*\*\*\*\*\*\***

Naredba 2.

Ovakav način unosa lozinke kao u naredbi 2, je povoljan kada se vriši pokretanje interaktivnih programa, međutim ukoliko se koristi neka skripta (ne interaktivni pristup) onda je neophodno postaviti lozinku unutar opcionog fajla. Fajl se nalazi u okviru operativnog sistema.

**[client]**

**password=password**

Kako bi se osigurao pristup fajlu samo korisniku koji ga je kreirao, potrebno je staviti status fajla na 400 ili 600.

**$> chmod 600 .my.cnf**

Naredba 3.

### **Logovanje i prikaz lozinki**

Lozinke se mogu napisati kao običan tekst u SQL izrazima kao što su CREATE USER, GRANT i SET PASSWORD, ili izrazima koji pozivaju funkciju PASSWORD(). Ako MySQL server evidentira ove upite onako kako su napisani, takve lozinke postaju dostupne svima koji imaju pristup evidenciji.

Evidentiranje izraza izbegava pisanje lozinki kao u samom tekstu za sledeće naredbe:

**CREATE USER ... IDENTIFIED BY ...**

**ALTER USER ... IDENTIFIED BY ...**

**SET PASSWORD ...**

**START SLAVE ... PASSWORD = ...**

**START REPLICA ... PASSWORD = ...**

**CREATE SERVER ... OPTIONS(... PASSWORD ...)**

**ALTER SERVER ... OPTIONS(... PASSWORD ...)**

Lozinke u ovim naredbama se prepisuju tako da se ne pojavljuju bukvalno u tekstu naredbe upisanom u opšti log upita, log sporih upita i binarni log. Prepisivanje se ne odnosi na druge naredbe. Konkretno, naredbe INSERT ili UPDATE za sistemsku tabelu mysql.user koje se odnose na lozinke se evidentiraju kao što jesu, tako da bi trebalo izbeći takve naredbe.

### **Akcije sa administratorkse strane za bezbednost lozinki**

MySQL skladišti lozinke za korisničke naloge u sistemskoj tabeli **mysql.user** . Pristup ovoj tabeli nikada ne bi trebalo da bude odobren nijednim neadministrativnim nalozima. Pored ovoga potrebno je postaviti odgovarajuće vreme trajanja lozinki, nakon čega se korisnicima šalje obaveštenje sa zahtevom da promene svoju staru lozinku.

[4]

## **Sistem kontrole pristupa i privilegija kod MySQL-a**

Sistem privilegija i kontrole pristupa je, ne samo kod MySQL-a već i kod svih drugih softvera koji su namenjeni za pristup bitnim podacima jedan od najbitnijih sistema za obezbeđivanje i povećanje bezbednosti. Kontorla pristupa je u neku ruku prva linija odbrane kod sistema baza podataka. Oni omogućavaju dodeljivanje različitih privilegija različitim vrstama korisnika. Pored različitih korisnika postoje i različite vrste privilegija koje oni nude.

### **Privilegije koje nudi MySQL**

MySQL pruža privilegije koje se primenjuju u različitim kontekstima i na različitim nivoima rada:

* Administrativne privilegije omogućavaju korisnicima da upravljaju radom MySQL servera. Ove privilegije su globalne jer nisu specifične za određenu bazu podataka.
* Privilegije baze podataka primenjuju se na bazu podataka i na sve objekte u njoj. Ove privilegije se mogu dodeliti za određene baze podataka ili globalno tako da se primenjuju na sve baze podataka.
* Privilegije za objekte baze podataka kao što su tabele, indeksi, prikazi i uskladištene rutine mogu se dodeliti za određene objekte u bazi podataka, za sve objekte datog tipa u bazi podataka (na primer, sve tabele u bazi podataka), ili globalno za sve objekti datog tipa u svim bazama podataka).

Dozvoljene privilegije za GRANT i REVOKE:

U nastavku je data tabela sa svim privilegijama.

|  |  |  |
| --- | --- | --- |
| **Privilege** | **Column** | **Context** |
| CREATE | Create\_priv | databases, tables, or indexes |
| DROP | Drop\_priv | databases, tables, or views |
| GRANT OPTION | Grant\_priv | databases, tables, or stored routines |
| LOCK TABLES | Lock\_tables\_priv | databases |
| REFERENCES | References\_priv | databases or tables |
| EVENT | Event\_priv | databases |
| ALTER | Alter\_priv | tables |
| DELETE | Delete\_priv | tables |
| INDEX | Index\_priv | tables |
| INSERT | Insert\_priv | tables or columns |
| SELECT | Select\_priv | tables or columns |
| UPDATE | Update\_priv | tables or columns |
| CREATE TEMPORARY TABLES | Create\_tmp\_table\_priv | tables |
| TRIGGER | Trigger\_priv | tables |
| CREATE VIEW | Create\_view\_priv | views |
| SHOW VIEW | Show\_view\_priv | views |
| ALTER ROUTINE | Alter\_routine\_priv | stored routines |
| CREATE ROUTINE | Create\_routine\_priv | stored routines |
| EXECUTE | Execute\_priv | stored routines |
| FILE | File\_priv | file access on server host |
| CREATE TABLESPACE | Create\_tablespace\_priv | server administration |
| CREATE USER | Create\_user\_priv | server administration |
| PROCESS | Process\_priv | server administration |
| PROXY | see proxies\_priv table | server administration |
| RELOAD | Reload\_priv | server administration |
| REPLICATION CLIENT | Repl\_client\_priv | server administration |
| REPLICATION SLAVE | Repl\_slave\_priv | server administration |
| SHOW DATABASES | Show\_db\_priv | server administration |
| SHUTDOWN | Shutdown\_priv | server administration |
| SUPER | Super\_priv | server administration |
| ALL [PRIVILEGES] |  | server administration |
| USAGE |  | server administration |

### **Privilegije i GRANT tabele**

Administratori sistema baze podataka kontrolišu i menjaju sadržaj GRANT tabela u MySQL-u indirektno koristeću već pomenute GRANT i REVOKE naredbe kako bi postavili naloge i privilegije dostupnih za svaki od njih.

U nastavku su prikazane MySQL tabele koje sadrže (GRANT) informacije o privilegijama:

* **user:** sadrži korisničke naloge, globalne privilegije i druge kolone bez privilegija.
* **global\_grants:** dinamičke globalne privilegije.
* **db:** sadrži privilegije na nivou baze podataka.
* **host:** nove MySQL verzije ne kreiraju više ovu tabelu od verzije 5.6.7.
* **tables\_priv:** sadrži privilegije na nivou tabele.
* **columns\_priv:** sadrži privilegije na nivou kolone.
* **procs\_priv:** sadrži privilegije uskladištenih procedura i funkcija.
* **proxies\_priv:** sadrži privilegije proksi (*eng.* proxy) korisnika.
* **default\_roles:** podrazumevane uloge korisnika.
* **role\_edges:** granice za podskup uloga.
* **password\_history:** istorija ažuriranja lozinki.

### **Specificiranje naloga i imena**

Imena MySQL naloga sastoje se od korisničkog imena i imena hosta. Ovo omogućava kreiranje naloga za korisnike sa istim imenom koji se mogu povezati sa različitih hostova. Ovim se definišu i opisuju prakse kako formirati imena naloga, uključujući posebne vrednosti i džoker pravila. U SQL izrazima kao što su CREATE USER, GRANT i SET PASSWORD, imena naloga se formiraju koristeći sledeća pravila:

* Sintaksa za imena naloga je sledećeg formata '**user\_name'@'host\_name**'.
* Ime naloga koje se sastoji samo od korisničkog imena je ekvivalentno 'user\_name'@'%'. Na primer, 'myusname' je ekvivalentno ' myusname '@'%'.
* Korisničko ime i ime hosta ne moraju biti pod navodnicima ako su legalni kao identifikatori bez navodnika. Navodnici su neophodni da se navede string user\_name koji sadrži specijalne znakove (kao što je „-”), ili niz host\_name koji sadrži specijalne znakove ili džoker znakove (kao što je „%“); na primer, 'test-user'@'%.com'.
* Navodite korisnička imena i imena hostova kao identifikatore ili stringove, koristeći pozadinske („`”), pojedinačne navodnike (“’”) ili dvostruke navodnike („“”).
* Delovi korisničkog imena i imena hosta, ako su navedeni, moraju se navesti odvojeno. To jest, potrebno je napisati 'me'@'localhost', a ne 'me@localhost'; ovo poslednje se tumači kao 'me@localhost'@'%'.
* Upućivanje na funkciju CURRENT\_USER ili CURRENT\_USER() je ekvivalentno doslovnom navođenju korisničkog imena i imena hosta trenutnog klijenta.

## **Kontrola pristupa**

Primarna funkcija MySQL sistema privilegija je da autentifikuje korisnika koji se povezuje sa datog hosta i da poveže tog korisnika sa privilegijama u bazi podataka kao što su SELECT, INSERT, UPDATE i DELETE. Dodatna funkcionalnost uključuje mogućnost davanja privilegija za administrativne operacije. Da bi kontrolisali koji korisnici mogu da se povežu, svakom nalogu se mogu dodeliti akreditivi za autentifikaciju, kao što je lozinka. Korisnički interfejs za MySQL naloge sastoji se od SQL naredbi kao što su CREATE USER, GRANT i REVOKE.

MySQL sistem privilegija osigurava da svi korisnici mogu obavljati samo one operacije koje su im dozvoljene. Kao korisnik, kada se poveže na MySQL server, njegov identitet je određen hostom sa kojeg se povezujete i korisničkim imenom koje navedete. Kada izda zahteve nakon povezivanja, sistem dodeljuje privilegije u skladu sa identitetom i onim što želi da radi.

MySQL uzima u obzir i ime hosta i korisničko ime da bi identifikovao korisnika jer nema razloga da se pretpostavi da dato korisničko ime pripada istoj osobi na svim hostovima. Na primer, korisnik joe koji se povezuje sa ‘office.example.com’ ne mora biti ista osoba kao korisnik joe koji se povezuje sa home.ekample.com. MySQL ovo rešava tako što vam omogućava da razlikujete korisnike na različitim hostovima koji slučajno imaju isto ime: možete dodeliti jedan skup privilegija za konekcije od strane korisnika sa office.example.com i drugačiji skup privilegija za konekcije od strane korisnika od kuće .example.com. Da biste videli koje privilegije ima dati nalog, koristite naredbu SHOW GRANTS. Na primer:

**SHOW GRANTS FOR 'joe'@'office.example.com';**

**SHOW GRANTS FOR 'joe'@'home.example.com';**

Naredba 4.

Interno, server skladišti informacije o privilegijama u tabelama odobrenja baze podataka mysql sistema. MySQL server čita sadržaj ovih tabela u memoriju kada se pokrene i zasniva odluke o kontroli pristupa na kopijama tabela odobrenja u memoriji. Kontrola pristupa MySQL-u uključuje dve faze kada pokrenete klijentski program koji se povezuje sa serverom.

### **Prva faza kontrole – verifikacija konekcije**

Kada korisnik pokušava da se poveže sa MySQL serverom, server prihvata ili odbija vezu na osnovu njegovog identiteta i toga da li možete da verifikujete svoj identitet davanjem ispravne lozinke. Ako ne, server vam potpuno odbija pristup. U suprotnom, server prihvata vezu, a zatim ulazi u fazu 2 i čeka na zahteve. Njegov identitet je zasnovan na dve informacije:

* Host klijenta sa kojeg se povezujete.
* Korisnikovo MySQL korisničko ime.

### **Druga faza kontrole – verifikacija zahteva**

Nakon što se uspostavi veza, server ulazi u fazu 2 kontrole pristupa. Za svaki zahtev koji se izda preko te veze, server određuje koju operaciju želite da izvršite, a zatim proverava da li ima dovoljne privilegije za to. Ovde stupaju u igru kolone privilegija u tabelama dodeljivanja. Ove privilegije mogu doći iz bilo koje tabele **user**, **db**, **tables\_priv**, **columns\_priv** ili **procs\_priv**.

Tabela korisnika daje privilegije koje su korisniku dodeljene na globalnoj osnovi i koje se primenjuju bez obzira na to koja je podrazumevana baza podataka. Na primer, ako korisnička tabela dodeli privilegiju DELETE, korisnik može izbrisati redove iz bilo koje tabele u bilo kojoj bazi podataka na hostu servera! Pametno je dodeliti privilegije u korisničkoj tabeli samo ljudima kojima su potrebne, kao što su administratori baze podataka. Tabela **db** daje privilegije specifične za bazu podataka. Vrednosti u kolonama opsega ove tabele mogu imati sledeće oblike:

* Prazna vrednost korisnika odgovara anonimnom korisniku. Vrednost koja nije prazna odgovara doslovno; u korisničkim imenima nema džoker znakova.
* Zamenski znakovi „%“ i „\_“ mogu se koristiti u kolonama Host i Db. Oni imaju isto značenje kao i za operacije podudaranja šablona koje se izvode sa LIKE operatorom.
* „%“ ili prazna vrednost Host znači „bilo koji host“.
* „%“ ili prazna Db vrednost znači „bilo koju bazu podataka“.

[7]

### **Dodeljivanje lozinki naloga**

Potrebni akreditivi za klijente koji se povezuju na MySQL server mogu uključivati lozinku. U MySQL 5.6, takođe je moguće da se klijenti autentifikuju pomoću dodataka.

Da bi dodelili lozinku kada kreiramo novi nalog sa CREATE USER, potrebno je uključiti klauzulu IDENTIFIED BY:

**$mysql> CREATE USER 'user'@'localhost' -> IDENTIFIED BY 'mypass';**

Naredba 5.

Da bi dodelili ili promenili lozinku za postojeći nalog, jedan od načina je da se izda naredba SET PASSWORD:

**$mysql> SET PASSWORD FOR -> 'user'@'localhost' = PASSWORD('mypass');**

Naredba 6.

## **SQL Injection kod MySQL-a i prevencija**

Problemi SQL injekcije se pojavljuju kada programeri softvera kreiraju dinamičke upite baze podataka odnosno konkatenaciju nizova koja uključuje unos koji daje korisnik. Izbeći greške u SQLinjekciji je jednostavno. Programeri treba da:

* prestanu da pišu dinamičke upite sa konkatenacijom nizova.
* sprečiti unos koji je dostavio korisnik koji sadrži zlonamerni SQL upit koji će uticati na logiku izvršenog upita.

[6]

**Primer ne bezbednog koda:** Sledeći primer (isečak 1) nije bezbedan i omogućio bi napadaču da ubaci kod u upit koji bi izvršila baza podataka. Nepotvrđeni parametar "**customerName**" koji se jednostavno dodaje upitu omogućava napadaču da ubaci bilo koji SQL kod koji želi. Nažalost, ovaj metod za pristup bazama podataka je previše uobičajen.

**String query = "SELECT account\_balance FROM user\_data WHERE username = "**

**+ request.getParameter("customerName");**

**try {**

**Statement statement = connection.createStatement( ... );**

**ResultSet results = statement.executeQuery( query );**

**}**

**...**

Isečak 1. (Java)

Kako bi se ovaj problem izbegao najlakše je uključiti već postojeće biblioteke za validiranje ovih slučajeva. Ispravan način da se izbegnu napadi SQL injekcije, bez obzira koja baza podataka se koristi, jeste da se odvoje podaci od SQL-a, tako da podaci ostaju podaci i da ih SQL parser nikada neće tumačiti kao komande. Moguće je kreirati SQL naredbu sa pravilno formatiranim delovima podataka, ali ako nismo sigurni ili ne razumemo konkretnu problematiku, uvek treba koristiti pripremljene izraze i parametrizovane upite. Ovo su SQL izrazi koje server baze podataka šalje i analizira odvojeno od svih parametara. Na ovaj način je nemoguće da napadač ubaci zlonamerni SQL. Ove biblioteke implementiraju različite funkcionalnosti koje vrše validaciju i unos parametrizovanih upita, i postoje različite vrste za različite programske jezike. A neke od najpopularnijih su:

* Java EE – koristite **PreparedStatement()** sa povezanim promenljivima.
* .NET – koristite parametrizovane upite kao što su **SqlCommand()** ili **OleDbCommand()** sa povezanim promenljivima.
* PHP – koristite PDO sa jako tipiziranim parametrizovanim upitima (koristeći **bindParam()**).
* Hibernate – koristite **createQuery()** sa promenljivim vezanja (koji se kod Hibernate-a nazivaju imenovani parametri).
* SQLite – koristite **sqlite3\_prepare()** za kreiranje objekta naredbe.

U ovom primeru (Isečak 2) vidimo pravilno napisan programski kod za prihvatanje SQL upita bez opasnosti od napada SQL injekcije. Kao najjednostavnije i pouzdanije rešenje je korišćenje postojeće biblioteke sa funkcijom prepareStatement(…) koja pre izvršavanje upita vrši njegovu validaciju i sprečava bilo kakav napad ili maliciozni upit.

**String customerName = request.getParameter("customerName");**

**String query = "SELECT account\_balance FROM user\_data WHERE username = ?";**

**PreparedStatement pstmt = connection.prepareStatement( query );**

**pstmt.setString( 1, customerName);**

**ResultSet results = pstmt.executeQuery( );**

**...**

Isečak 2. (Java) [6]

Ukoliko koristimo MySQL komand interfejs, sledeći primer (Isečak 3) to pokazuje:

**$stmt = $dbConnection->prepare('SELECT \* FROM user\_data WHERE username = ?');**

**$stmt->bind\_param('s', $username); // 's' definiše tip parametra => 'string'**

**$stmt->execute();**

**$result = $stmt->get\_result();**

**while ($row = $result->fetch\_assoc()) {**

**// rad sa redovima => $row**

**}**

Isečak 3. (MySQLi)

## **Enkripcija podataka**

MySQL enkripcija je proces šifrovanja baze podataka koji praktikuje transformaciju običnog teksta i tekstualnih zapisa podataka u bazi podataka servera u nerazumljiv heširani tekst uz pomoć algoritma za šifrovanje. Generalno, kada bilo koji korisnik pregleda podatke u bazi podataka, tada će podaci zasnovani na tipu kolone tabele biti čitljivi kao tekst. Nekoliko klasično korišćenih kategorija polja za kolone u tabeli baze podataka kao što su VARCHAR, INT, TEXT i DATETIME su u ljudskom čitljivom obliku, osim blob-ova i nekih drugih kategorija polja.

Dakle, MySQL enkripcija se izvodi ili šifrovanjem kontejnera koji sadrži pohranjene MySQL zapise podataka, na primer memoriju diska, ili šifrovanjem podataka podataka pre nego što ih usmeri na MySQL server.

### **Postavljanje enkripcije**

Funkcije MySQL Enterprise Encription nalaze se u datoteci biblioteke funkcija koja se može učitati, instaliranoj u direktorijumu dodataka (direktorijumu koji je nazvan sistemskom promenljivom **plugin\_dir**). Osnovno ime biblioteke funkcija je **openssl\_udf**, a sufiks zavisi od platforme. Na primer, ime datoteke na Linux-u ili Windows-u je **openssl\_udf.so** ili **openssl\_udf.dll**, respektivno. Da bi instalirali funkcije iz datoteke biblioteke, koristite naredbu CREATE FUNCTION. Da bi učitali sve funkcije iz biblioteke, koristi se ovaj skup upita, prilagođavajući sufiks imena datoteke po potrebi:

**CREATE FUNCTION asymmetric\_decrypt RETURNS STRING**

**SONAME 'openssl\_udf.so';**

**CREATE FUNCTION asymmetric\_derive RETURNS STRING**

**SONAME 'openssl\_udf.so';**

**CREATE FUNCTION asymmetric\_encrypt RETURNS STRING**

**SONAME 'openssl\_udf.so';**

**CREATE FUNCTION asymmetric\_sign RETURNS STRING**

**SONAME 'openssl\_udf.so';**

**CREATE FUNCTION asymmetric\_verify RETURNS INTEGER**

**SONAME 'openssl\_udf.so';**

**CREATE FUNCTION create\_asymmetric\_priv\_key RETURNS STRING**

**SONAME 'openssl\_udf.so';**

**CREATE FUNCTION create\_asymmetric\_pub\_key RETURNS STRING**

**SONAME 'openssl\_udf.so';**

**CREATE FUNCTION create\_dh\_parameters RETURNS STRING**

**SONAME 'openssl\_udf.so';**

**CREATE FUNCTION create\_digest RETURNS STRING**

**SONAME 'openssl\_udf.so';**

Naredba 7.

### **Mogućnosti kod enkripcije podataka**

MySQL Enterprise Edition (*eng.* poslovna verzija sistema beze podataka) uključuje skup funkcija šifrovanja odnosno enkriptovanja zasnovanih na OpenSSL biblioteci koje otkrivaju mogućnosti OpenSSL-a na SQL nivou. Ove funkcije omogućavaju Enterprise aplikacijama da obavljaju sledeće operacije:

* Implementiranje dodatne zaštite podataka koristeći asimetričnu kriptografiju sa javnim ključem
* Kreiranje javnih i privatnih ključeva i digitalne potpise
* Izvršenje asimetričnog šifrovanja i dešifrovanje
* Kriptografsko heširanje za digitalno potpisivanje i verifikaciju i validaciju podataka
* MySQL Enterprise Encription podržava RSA, DSA i DH kriptografske algoritme.
* MySQL Enterprise Encription se isporučuje kao biblioteka funkcija koje se mogu učitati, iz kojih se pojedinačne funkcije mogu pojedinačno instalirati.

MySQL Enterprise Enkripcija pruža funkcije tipične za industriju za asimetrično šifrovanje za zaštitu osetljivih podataka tokom njegovog životnog veka, tj. kriptografiju javnog ključa. MySQL Enterprise Encription je odgovoran za šifrovanje, digitalne potpise, generisanje ključeva i neke kriptografske karakteristike za uslužne organizacije koje štite intimne podatke i ispunjavaju regulatorne potrebe koje obuhvataju Sarbanes-Oklei, HIPAA i PCI standard bezbednosti podataka.

Takođe, MySQL Enterprise Encription obezbeđuje DBA-ima kao i programerima tehničke alate koji su im potrebni za dole navedene:

* Asimetrično šifrovanje javnog ključa
* Asimetrično dešifrovanje javnog ključa
* Generisanje javnog/privatnog ključa
* Obezbeđivanje simetričnih ključeva iz parova javnih i privatnih ključeva
* Digitalni potpisivanje podataka
* Proveravanje potpisa podataka
* Autentifikovanje podataka za validaciju

### **Kako funkcioniše enkripcija kod MySQL-a**

InnoDB tabele u MySQL-u održavaju enkripciju podataka u mirovanju za opšte prostore tabela. U MySQL 8.0.16, da bi se podrazumevano podesilo šifrovanje za DBMS šeme i slično, takođe se održavaju opšti prostori tabela koji omogućavaju DBA-ima da nadgledaju da li su tabele proizvedene u tim šemama i povezanim prostorima tabela kodirani.

Svojstvo šifrovanja podataka u mirovanju zavisi od dodatka za ključeve koji služe za upravljanje glavnim ključem šifrovanja. Ovaj dodatak za datoteku ključeva je obezbeđen za sva izdanja MySQL-a gde su podaci o ključevima smešteni u datoteci koja je lokalna za host servera. MySQL Enterprise Edition isporučuje dodatni dodatak za ključeve:

* Keyring\_encrypted\_file čuva podatke ključeva unutar šifrovane datoteke lokalno za host servera.
* Keyring\_okv sadrži KMIP klijenta, tj. KMIP 1.1, koji implementira KMIP-kompatibilan proizvod za skladištenje ključeva koji radi u pozadini (back-end).
* Keyring\_aws se povezuje sa amazon web servisima (Amazon Web Services – AWS) Key Management Service (KMS) za generisanje ključeva kao pozadinu i za skladištenje ključeva primenjuje lokalnu datoteku.
* Keyring\_hashicorp se pridružuje HashiCorp trezoru (*eng.* vault) za skladištenje pozadinskog dela (back-end-a).

### **Primeri enkripcije**

U MySQL 8.0.16 verziji, promenljiva pod nazivom **default\_table\_encription** reguliše šifrovanje neposredno generisanog prostora tabele osim klauzule **ENCRIPTION** koja je otvoreno identifikovana u upitu **CREATE** **TABLESPACE**. U prethodnoj verziji MySQL 8.0.16, trebalo bi navesti klauzulu **ENCRYPTION** da bi se omogućilo šifrovanje kao što je napisano u nastavku (naredba 8):

**CREATE TABLESPACE ‘tbs1’ ADD DATAFILE ‘tbs.ibd’ ENCRYPTION = ‘Y’ ENGINE=InnoDB;**

Naredba 8.

Za modifikaciju šifrovanja trenutnog opšteg prostora tabele, potrebno je navesti klauzulu **ENCRIPTION**:

**ALTER TABLESPACE tbs ENCRYPTION = ‘Y’;**

Naredba 9.

Za šifrovanje MySQL sistemskog tabelarnog prostora, MySQL 8.0.16 ga takođe podržava. Ovaj MySQL sistemski tabelarni prostor sadrži MySQL sistemsku bazu podataka zajedno sa MySQL tabelama rečnika podataka i podrazumevano je nešifrovan. Dakle, da bi se omogućilo šifrovanje, mora se identifikovati ime prostora tabele kao i opciju **ENCRYPTION** u okviru **ALTER** **TABLESPACE** iskaza kao što je prikazano u nastavku:

**ALTER TABLESPACE mysql ENCRYPTION = ‘Y’;**

Naredba 10.

Takođe, da bismo ograničili šifrovanje za ovaj mysql sistemski prostor tabele, moramo da postavimo **ENCRIPTION = 'N'** primenom naredbe upita **ALTER** **TABLESPACE**:

**ALTER TABLESPACE mysql ENCRYPTION = ‘N’;**

Naredba 11.

Kad god je opcija **ENCRYPTION** navedena u upitu, **CREATE** **TABLE** ili **ALTER** **TABLE**, ona je dokumentovana u koloni tabele **CREATE\_OPTIONS** **INFORMATION\_SCHEMA.TABLES**. Možemo proveriti kolone kako bi validirali tabele koje ostaju šifrovane u prostoru tabela.

**SELECT TABLESCHEMA, TABLENAME, CREATE\_OPTIONS FROM INFORMATION\_SCHEMA.TABLES WHERE CREATE \_OPTIONS LIKE ‘%ENCRYPTION’;** Naredba 12.

[8]

## **Rezervne kopije i povratak podataka**

Mnogo puta problem sa upadima u bazu podataka nisu samo krađa podataka, već se često javljaju slučajevi gde je cilj upada u bazu, nanošenje štete samom korisniku baze, što uljucuje brisanje i uništavanje podataka. Jedan od najboljih načina za zaštitu od ovih problema jeste pravljenje redovne rezervne kopije (*eng.* backup) kao i pravljenje kopija koje nisu direktno vezane na istu mrežu sa glavnom bazom (ukoliko se radi o sistemu baza podataka).

Uslužni program **mysqldump** klijent može da izbaci bazu podataka uključujući SQL upite potrebne za ponovnu izgradnju (*eng.* rebuild) baze podataka.

Podrazumevano, dump datoteka uključuje SQL komande za vraćanje tabela i podataka.

**Kreiranje rezervne kopije pomoću mysqldump-a**

Da bi napravili rezervnu kopiju vaše MySQL baze podataka, opšta sintaksa je:

**sudo mysqldump -u [user] -p [database\_name] > [filename].sql**

Naredba 13.

* [**user**] korisničko ime i lozinka(ako je potrebno).
* [**database\_name**] je putanja i ime datoteke baze podataka.
* Komanda **>** specificira izlaz.
* [**filename**] je putanja i ime datoteke u kojoj želite da sačuvate datoteku **damp**-a.

Da bi napravili rezervnu kopiju čitavog sistema za upravljanje bazom podataka:

**mysqldump --all-databases --single-transaction --quick --lock-tables=false > full-backup-$(date +%F).sql -u root -p**

Naredba 14.

Da bi uključili više od jedne baze podataka u datoteku rezervne kopije:

**sudo mysqldump -u [user] -p [database\_1] [database\_2] [database\_etc] > [filename].sql**

Naredba 15.

**Povratak baze iz rezervne kopije pomoću mysqldump-a**

Na sistemu koji hostuje bazu podataka, pomoću MySQL-a kreirati novu bazu podataka.

Sada je potrebno uveriti se da ste ga nazvali isto kao i baza podataka koju ste izgubili. Ovo kreira osnovnu datoteku u koju će mysqldump uvesti podatke. Pošto dump datoteka ima komande za ponovnu izgradnju baze podataka, potrebno je samo da kreirati praznu bazu podataka.

Sledećom komandom vršimo restauraciju baze podataka:

**mysql -u [user] -p [database\_name] < [filename].sql**

Naredba 16.

Obavezno uključiti[**database\_name**] i [**filename**] u putanju. Verovatno je da na glavnoj mašini [**database\_name**] može biti u osnovnom direktorijumu, tako da možda neće biti potrebno da se dodaje putanja. Nakon toga potrebno je uveriti se da je navedena tačna putanja za dump datoteku koja se vraća, uključujući ime servera (ako je potrebno).

## **Sigurnost baza podataka sa strane infrastrukture**

Pored postavljanja samih bezbednosnih mehanizama koje MySQL nudi, veliki deo sigurnosti i stabilnosti baze podataka se odnosi na samu infrastrukturu koja igra ulogu podrške same baze podataka ([my]SQL server). Veliku ulogu nosi server jer on ima ulogu da održava stabilnost baze podataka, sprečava DDoS napade, kao i potrebu za razne bezbednostne zaštite / štitove (*eng.* Firewall).

### **DDoS napadi**

**DDoS** napad znači (*eng.* Distributed Denial-of-Service) napad i to je sajber zločin u kojem napadač pokušava da preplavi server internet saobraćajem kako bi sprečio korisnike da pristupe povezanim onlajn uslugama i konkretno serveru MySQL baze podataka. Motivacije za sprovođenje DDoS-a su veoma različite, kao i vrste pojedinaca i organizacija koje žele da izvrše ovaj oblik sajber napada. Neke napade izvode nezadovoljni pojedinci i haktivisti koji žele da uklone servere kompanije samo da bi dali izjavu, zabavili se iskorišćavanjem sajber slabosti ili izrazili neodobravanje. [12]

MySQL nudi neka od rešenja za ovaj problem a oni se svode na ograničavanje upotrebu resursa od strane korisnika po jedinici vremena. Ograničenja koja nudi su:

* Broj upita koji nalog može da izda po satu
* Broj ažuriranja koje nalog može da izda po satu
* Koliko puta se nalog može povezati sa serverom po satu
* Broj istovremenih veza sa serverom preko naloga

Pre MySQL 5.0, „nalog“ je vezivan u odnosu na stvarni host sa kojeg se korisnik povezuje. Ovaj stariji metod obračuna se može izabrati pokretanjem servera sa opcijom **--old-stile-user-limits**. U ovom slučaju, ako se korisnik istovremeno povezuje sa **host1.example.com** i **host2.example.com**, server primenjuje ograničenja resursa naloga posebno na svaku vezu. Ako se korisnik ponovo poveže sa **host1.example.com**, server primenjuje ograničenja za tu vezu zajedno sa postojećom vezom sa tog hosta.

Da bi postavili ograničenja resursa za nalog, koristite se izjava GRANT. Potrebno je obezbediti klauzulu WITH koja imenuje svaki resurs koji treba da bude ograničen. Podrazumevana vrednost za svako ograničenje je nula (bez ograničenja). Na primer, da bi kreirali novi nalog koji može da pristupi bazi podataka klijenata, ali samo na ograničen način, izdaju se ove izjave (naredba 17):

**$mysql> CREATE USER 'michael'@'localhost' IDENTIFIED BY 'mich';**

**$mysql> GRANT ALL ON customer.\* TO 'michael'@'localhost'**

**WITH MAX\_QUERIES\_PER\_HOUR 20**

**MAX\_UPDATES\_PER\_HOUR 10**

**MAX\_CONNECTIONS\_PER\_HOUR 5**

**MAX\_USER\_CONNECTIONS 2;**

Naredba 17.

Ovom naredbom postavljamo limite za svaku vrstu pristupa ponaosob. Ne moraju svi tipovi ograničenja biti imenovani u klauzuli WITH, ali oni koji su imenovani mogu biti prisutni bilo kojim redosledom. Vrednost za svako ograničenje po satu treba da bude ceo broj koji predstavlja broj po satu. Za MAX\_USER\_CONNECTIONS, ograničenje je ceo broj koji predstavlja maksimalan broj istovremenih veza naloga. Ako je ovo ograničenje postavljeno na nulu, globalna vrednost sistemske promenljive max\_user\_connections određuje broj istovremenih veza. Ako je max\_user\_connections takođe nula, nema ograničenja za nalog.

Da bi modifikovali ograničenja za nalog, koristimo upit **GRANT USAGE** na globalnom nivou **(ON \*.\*)**. Sledeći uput menja ograničenje upita za korisnika **michael** na 100.

$mysql> GRANT USAGE ON \*.\* TO 'michael'@'localhost' WITH MAX\_QUERIES\_PER\_HOUR 100;

Naredba 18.

Upit menja samo navedenu graničnu vrednost i ostavlja nalog nepromenjenim. Da bi uklonili ograničenje, postavljamo njegovu vrednost na nulu. Na primer, da bi uklonili ograničenje koliko puta na sat Michael može da se poveže, koristimo ovaj upit:

$**mysql> GRANT USAGE ON \*.\* TO 'michael'@'localhost'**

**WITH MAX\_CONNECTIONS\_PER\_HOUR 0;**

Naredba 19.

Kao što je ranije pomenuto, ograničenje simultane veze za nalog se određuje iz ograničenja MAX\_USER\_CONNECTIONS i sistemske promenljive max\_user\_connections. Pretpostavimo da je globalna vrednost max\_user\_connections 10 i da tri naloga imaju pojedinačna ograničenja resursa navedena na sledeći način:

**GRANT ... TO 'user1'@'localhost' WITH MAX\_USER\_CONNECTIONS 0;**

**GRANT ... TO 'user2'@'localhost' WITH MAX\_USER\_CONNECTIONS 5;**

**GRANT ... TO 'user3'@'localhost' WITH MAX\_USER\_CONNECTIONS 20;**

Naredba 20.

**user1** ima ograničenje veze od 10 (globalna vrednost max\_user\_connections) jer ima ograničenje MAX\_USER\_CONNECTIONS od nula. user2 i user3 imaju ograničenja veze od 5 i 20, respektivno, jer imaju različita od nula ograničenja od MAX\_USER\_CONNECTIONS.

Server čuva ograničenja resursa za nalog u redu korisničke tabele koji odgovara nalogu. Kolone max\_questions, max\_updates i max\_connections čuvaju ograničenja po satu, a kolona max\_user\_connections čuva ograničenje MAX\_USER\_CONNECTIONS. Brojanje upotrebe resursa se dešava kada bilo koji nalog ima ograničenje koje nije nula za korišćenje bilo kog od resursa. Dok server radi, on broji koliko puta svaki nalog koristi resurse. Ako nalog dostigne ograničenje broja veza u poslednjem satu, server odbija dalje veze za nalog dok taj sat ne istekne. Slično tome, ako nalog dostigne ograničenje broja upita ili ažuriranja, server odbija dalje upite ili ažuriranja dok ne istekne sat. U svim takvim slučajevima, server izdaje odgovarajuće poruke o grešci.

[14]

### **Replikacija MySQL baze podataka**

Kao poslednji deo u ovom radu kada je u pitanju sigurnost MySQL baze podataka je replikacija same baze podataka.

MySQL replikacija je proces koji omogućava da se podaci sa jednog MySQL servera baze podataka (master) automatski kopiraju na jedan ili više MySQL servera baze podataka (slave). Obično se koristi za širenje pristupa za čitanje na više servera radi skalabilnosti, iako se može koristiti i za druge svrhe, kao što je prelazak na grešku ili analiziranje podataka na slave-u kako se ne bi preopteretio master. Pošto je replikacija master-slave jednosmerna replikacija (od master-slave-a), samo se glavna baza podataka koristi za operacije pisanja, dok se operacije čitanja mogu proširiti na više podređenih baza podataka. Ovo znači da ako se replikacija master-slave koristi kao rešenje za skaliranje, mora se imati najmanje dva definisana izvora podataka, jedan za operacije pisanja, a drugi za operacije čitanja. [15]

**Primer**: u ovom slučajju zahtevalo bi se kreiranje najmanje dve instance baze podataka od kojih bi jedna bila master druga slave, i gde bi se vršila replikacija mastera na slave-u.

Kada imamo dve MySQL instance koje rade na istoj mašini, postavićemo prvu instancu kao glavnu, a drugu kao slave. Jedina preostala promena u datoteci **my.cnf** je postavljanje binarnog evidentiranja na master. Da bi smo to uradili, potrebno je urediti **my.cnf** datoteku sa sledećim izmenama i dodacima u grupi [**mysqld1**]:

**log\_bin                     = /var/log/mysql/mysql-bin.log**

**innodb\_flush\_log\_at\_trx\_commit  = 1**

**sync\_binlog                 = 1**

**binlog-format               = ROW**

Naredba 21.

Nakon ovoga potrebno je izvršiti restart glavne instance. Da bi se slave mogao povezati sa masterom sa ispravnim privilegijama replikacije, na masteru treba kreirati novog korisnika. Potrebno je povezati se na glavnu instancu koristeći MySQL klijent sa odgovarajućim hostom i portom:

**mysql -uroot -p --host=127.0.0.1 --port=3306**

Nakon ovoga vrši se kreiranje novog korisnika za replikaciju:

**$mysql> CREATE USER 'replication'@'%' IDENTIFIED BY 'replication';**

**$mysql> GRANT REPLICATION SLAVE ON \*.\* TO 'replication'@'%';**

Sada je potrebno izvršiti komandu kojom se kreira dump podaci mastera:

**$> mysqldump -uroot -p --host=127.0.0.1 --port=3306 --all-databases --master-data=2 > replicationdump.sql**

Drugi deo ovog procesa predstavlja unošenje dump-a u slave instancu:

**mysql -uroot -p --host=127.0.0.1 --port=3307 < replicationdump.sql**

Konfigurisanje master-slave replikacije u razvojnom okruženju je korisno ako je potrebna za rešenje za skaliranje u proizvodnom okruženju. Na ovaj način ćemo takođe imati odvojene izvore podataka konfigurisane za operacije pisanja i čitanja tako da možete lokalno testirati da li sve funkcioniše kako se očekuje pre daljeg postavljanja.

# **Zaključak**

Osigurati da imamo najviši nivo bezbednosti na mreži je ključno, za nesmetano i pouzdano funkcionisanje baze podataka u konstatno razvijajućim sredinama. Bezbednost baze podataka može značiti razliku između uspesnog i neuspesnog poslovanja kompanija kao i poverljivost podataka sa kojima komanije raspolažu.

Baze podataka se danas upotrebljavaju u ogromnim merama, a to važi i za samu MySQL bazu podataka i njegov sistem, pa se i potreba za njihovo upotrebljavanje povećava. Ovakve vrste baze podataka, poseduju veliku poslovnu reputaciju što im i obezbeđuje stalan rast korisnika i poslovanja, pa je stoga i potreban konstantan razvoj bezbednosnih aspekata sistema, jer je bezbendnost korisničkih podataka od krucijalnog značaja.

MySQL baza podataka kako je navedeno u samom radu nudi razne vrste mehanizama za zaštitu podataka i pristupa bazi. Glavni fokus je stavljen na privilegije i kontrolu pristupa različitim vrstama korisnika baze podataka kako bi se postavila jasna hijerarhija kad su upitanju privilegije. Pored toga postoje i različite prakse koje se prate kako bi se sama sigurnost baze podataka ispoštovala. Stoga možemo reći da je sistem kontrole pristupa MySQL-a veoma dobar sistem koji nudi mnogo mogućnosti uorganizaciji rada i upotrebe baze podataka.

Posebnu pažnju treba obratiti na pisanje samih softvera koji će raditi sa bazama podataka kako bi se ispoštovale sve prakse koje postoje kako bi se izbegli zlonamerni napadi u obliku SQL injekcije ili neke druge vrste.

MySQL baza podataka konstantno raste i razvija se, pa se samim tim uvode novi sistemi za obezbeđivanje sistema, pored toga napadači takođe konstantno nalaze nove načine kako bi izvršili upad pa je sama tematika napadača i sistema odbrane usko povezana jer razvoj jedne grane IT industrije uvek povlači napredak druge strane i obrnuto, tako da ovo predstavlja deo sistema koji će se najviše razvijati i koji će najviše ulivati poverenje kod korisnika, jer sami klijenti bezobzira na količinu funkcionalnosti sistem nudi, neće se opredeliti za njega ukoliko je bezbednost sistema neprihvatljiva njihovim zahtevima.

# **Literatura**

[1] [What is Database Security | Threats & Best Practices | Imperva](https://www.imperva.com/learn/data-security/database-security/) - <https://www.imperva.com/learn/data-security/database-security/>

[2] [MySQL :: MySQL 8.0 Reference Manual :: 6 Security](https://dev.mysql.com/doc/refman/8.0/en/security.html) - <https://dev.mysql.com/doc/refman/8.0/en/security.html>

[3] [MySQL Security - w3resource](https://www.w3resource.com/mysql/mysql-security.php) - <https://www.w3resource.com/mysql/mysql-security.php>

[4] [MySQL :: MySQL 8.0 Reference Manual :: 6.1.2 Keeping Passwords Secure](https://dev.mysql.com/doc/refman/8.0/en/password-security.html) - <https://dev.mysql.com/doc/refman/8.0/en/password-security.html>

[5] [9 Advanced MySQL Security Tips (makeuseof.com)](https://www.makeuseof.com/mysql-advanced-security-tips/) - <https://www.makeuseof.com/mysql-advanced-security-tips/>

[6] [SQL Injection Prevention - OWASP Cheat Sheet Series](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html) - <https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html>

[7] [MySQL :: MySQL 8.0 Reference Manual :: 6.2 Access Control and Account Management](https://dev.mysql.com/doc/refman/8.0/en/access-control.html) - <https://dev.mysql.com/doc/refman/8.0/en/access-control.html>

[8] [MySQL encryption | How does encryption work in MySQL with examples? (educba.com)](https://www.educba.com/mysql-encryption/) - <https://www.educba.com/mysql-encryption/>

[9] [How to Encrypt Entire MySQL Database (idatamigration.com)](https://www.idatamigration.com/data_migration_tools/encrypt_mysql_database.php#howToEncryptAllMysqlDatabaseTables) - <https://www.idatamigration.com/data_migration_tools/encrypt_mysql_database.php#howToEncryptAllMysqlDatabaseTables>

[11] [How to Back Up and Restore a MySQL Database {Easy Tutorial} (phoenixnap.com)](https://phoenixnap.com/kb/how-to-backup-restore-a-mysql-database) - <https://phoenixnap.com/kb/how-to-backup-restore-a-mysql-database>

[12] [What is a DDoS Attack? DDoS Meaning, Definition & Types | Fortinet](https://www.fortinet.com/resources/cyberglossary/ddos-attack) - <https://www.fortinet.com/resources/cyberglossary/ddos-attack>

[13] [security - Methods to prevent DDoS based MySQL queries submitted through URL string - Stack Overflow](security%20-%20Methods%20to%20prevent%20DDoS%20based%20MySQL%20queries%20submitted%20through%20URL%20string%20-%20Stack%20Overflow) - <https://stackoverflow.com/questions/19496097/methods-to-prevent-ddos-based-mysql-queries-submitted-through-url-string>

[14] [MySQL :: MySQL 5.6 Reference Manual :: 6.2.13 Setting Account Resource Limits](https://dev.mysql.com/doc/refman/5.6/en/user-resources.html) - <https://dev.mysql.com/doc/refman/5.6/en/user-resources.html>

[15] [MySQL Master-Slave Replication Tutorial | Toptal](https://www.toptal.com/mysql/mysql-master-slave-replication-tutorial) - <https://www.toptal.com/mysql/mysql-master-slave-replication-tutorial>