Elektronski fakultet u Nišu

Seminarski rad

PostgreSQL - Sigurnost baze podataka

Predmet: Sistemi za upravljanje bazama podataka

|  |  |
| --- | --- |
| Mentor: Aleksandar Stanimirović | Student: Marko Đokić 1022 |

*April, 2020.*

Sadržaj

[Sigurnost baze podataka](#_q6qesbyualhl)

[Identifikacija korisnika](#_rsfrc28d4zmy)

[Kontrola pristupa zasnovana na hostu](#_j58vz7n5303p)

[Metode autentifikacije](#_c5p28f47t8fp)

[Najbolje prakse autentifikacije](#_vwgzdse4zibt)

[Sistem uloga](#_ndytaotzedn9)

[Nasleđivanje grupnih i korisničkih uloga](#_qcvcnwwdqk0u)

[Uklanjanje uloga](#_twvz5qnwzf0o)

[Zaštita na nivou reda](#_4ez05f61r00j)

[Zaključak](#_q9wjyj4eqx2x)

[Literatura](#_8le5ci3l51nq)

# Sigurnost baze podataka

Sigurnost i zaštita podataka su ključni za održavanje i kvalitet poslovanja. Zaštita podataka nije podrazumevana, ali je neophodna da bi sistem bio dobar, legalan i kvalitetan. Osetljive podatke kao što su korisnički podaci, e-mail adrese, geografske adrese i podatke o plaćanju trebalo bi zaštititi od bilo kakvog ilegalnog pristupa.

Postoji nekoliko nivoa zaštite podataka, koji se definišu kod politike o zaštiti podataka i državnim zakonima. Politika o zaštiti podataka, obično definiše način širenja podataka različitim stranama, korisnicima sa ovlašćenim pristupom i tako dalje. Podaci treba da budu zaštićeni na različitim nivoima, uključujući prenos i šifrovanje podataka u uređaju za skladištenje. Sigurnost podataka je velika teme i često postoje u kompanijama inženjeri (menadžeri) za sigurnost podataka posvećeni samo tim zadacima.

Sigurnost i bezbednost baze podataka kod PostgreSQL baze podataka se na nekoliko nivoa:

* Zaštita datoteka baze podataka. Sve datoteke sačuvane u bazi podataka zaštićene su od čitanja od strane bilo koga osim PostgreSQL superuser naloga.
* Veze klijenta sa serverom baze podataka su, podrazumevano, dozvoljene samo preko lokalnog Unix soketa, a ne preko TCP/IP soketa. Bekend se mora započeti opcijom -i kako bi se klijenti koji nisu lokalni povezali sa bazom podataka.
* Klijentske veze mogu biti ograničene IP adresama i/ili pomoću korisničkih imena direktno u pg\_hba.conf fajlu u PG\_DATA delu.
* Klijentske veze mogu sadržati autentifikaciju i prema neki eksternim paketima.
* Svakom korisniku PostgreSQL baze dodeljeno je korisničko ime i (opciono) lozinka. Korisnici podrazumevano nemaju pristup pisanja u baze podataka koje nisu kreitrali.
* Korisnici se mogu dodeliti grupama, a pristup tabeli može biti ograničen na osnovu grupnih privilegija iz fajla privileges.tz. (1)(2)

## Identifikacija korisnika

Za sve korisnike koji pokušaju da pristupe PostgreSQL proverava se u sadržaju klase pg\_user da li su ovlašćeni za to. Međutim, verifikacija stvarnog identiteta korisnika se vrši na različite načine:

* Iz korisničkog šela (user shell)

Bekend server pokrenut od strane korisničkog šela beleži korisnikov (efektivni) id pre nego što se izvrši operacija setovanja id za tog korisnika. Korisnički efektivni id se koristi kao osnovna provera kontrole pristupa, ne vrši se druga autentifikacija.

* Sa mreže

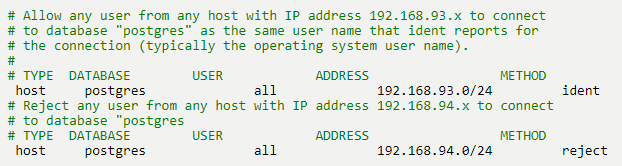
Ako je PostgreSQL sistem napravljen kao distribuiran, pristup preko TCP porta glavnom procesu je dostupan svakome. Baza konfiguriše datoteku pg\_hba.conf u PGDATA direktorijumu koja određuje koji će se sistem provere autentičnosti koristiti za uspostavljanje veze sa hostom i sa kojom bazom podataka će se povezati. Naravno autentifikacija na bazi hosta takođe nije sigurna. Uljezi se mogu maskirati u izvornog hosta i ta pitanja bezbednosti su izvan opsega PostgreSQL. (1)

## Kontrola pristupa zasnovana na hostu

Kontrola pristupa zasnovana na hostu je naziv osnovne kontrole kod PostgreSQL, beleže se stvari vezane o tome koji klijenti mogu imati pristup bazi podataka i kako se korisnici tih klijenata moraju autentifikovati.

Kao što je već pomenuto svaki sistem sadrži datoteku pg\_hba.conf, u PGDATA direktorijumu, koja kontroliše kako se ko može povezati sa svakom bazom podataka.

Svaki klijent koji pristupa bazi mora biti pokriven jednom od opcija u pg\_hba.conf. U suprotnom će se svi pokušaji veza sa tim klijentom odbaciti sa porukom greške. Opšti format ove datoteke je skup zapisa po jedan po redu, a prazne linije i linije koje počinju karakterom taraba (“#”) se zanemariju. zapis je sastavljen od broja polja koja su odvojena razmacijma i/ili tabulatorima. Format datoteke prikazan na slici 1 u nastavku:



Slika 1. Format fatoteke pg\_hba.conf

Povezivanje se može uspostaviti pomoću Unix soketa ili internet soketa TCP/IP. Veze napravljene pomoću Unix soketa se kontorlišu zapisima sledećeg formata:

local ***database user*** ***authentication\_method***

gde ***user*** korisnik koji može pristupiti na ovaj način, ***database*** podatak se odnosi na bazu na koju se ovaj zapis odnosi. Da je tu stojalo ***all*** to bi se odnosilo na sve baze podataka ili sve korisnike. ***authentication method*** određuje metodu koju korisnik mora da koristi za proveru autentičnosti prilikom povezivanja na tu bazu pomoću Unix soketa. U nastavku rada će biti opisane različite metode autentifikacije.

Veze kreirane korišćenjem sokete internet domena kontorlišu se korišćenjem zapisa sledećeg formata:

host ***database user*** ***TCP/IP\_address***/***mask*** ***authentication\_method***

Umesto host može stojati i hostssl gde se konekcija šifruje sa SSL šifrovanjem i hostnossl gde veza ne treba da bude šifrovana.

Uz ***TCP/IP\_address*** dodaje se ***mask*** i to predstavlja TCP/IP opseg adresa klijenta. Ako se neka adresa odgovara adresi klijenta ovo će se koristiti za konekciju. Ako se adresa poklapa sa više opsega adresa, koristi se onaj koji je prvi od tih u datoteci. ***TCP/IP\_address*** adresa se zapisuju u decimalnom zapisu sa tačkama.

Ako se adresa ne podudara ni sa jedom, primenjuje se odbijanje konekcije. (1)

## Metode autentifikacije

Sledeće metode autentifikacije podržane su kod oba Unix i TCP/IP soketa:

***trust***

* Veza je dozvoljena bezuslovno.

***reject***

* Veza se bezuslozno obdija.

***ident***

* Pribavlja se korisničko ime od operativnog sistema klijenta i koristi se za pristup bazi. Ovaj način se preporučuje kod zatvorenih mreža, gde su klijentske mašine strogo kontrolišu od strane administratora sistema.

***peer***

* Funkcioniše slično kao ***ident,*** ali se korisničko ime klijentovog operativnog sistema dobija iz kernela.

***crypt***

* Klijentu se traži lozinka za korisnika. To se šalje šifrovano i upoređuje se sa lozinkom pg\_shadow tabele. Ako se lozinke podudaraju veza je dozvoljena.

***password***

* Klijentu se traži lozinka za korisnika. Šalje se jasno i upoređuje sa lozinkom u tabeli pg\_shadow. Ako se lozinke podudaraju, veza je dozvoljena. Opciono password fajl može se dodati nakon ključne reči password, koji se koristi umesto pg\_shadow.

***scram-sha-256***

* Skoro najjača metoda koja je uvedena u PostgreSQL 10. Sprečava njuškanje lozinke na nepouzdanim vezama.

***md5***

* Manje siguran metod od prethodnog i sprečava njuškanje lozinke u mreži, ali ne pruža zaštitu ako napadač uspe da ukrade heš lozinke sa servera.

Postoje i slične metode kao što su ***GSSAPI*** i ***LDAP.***

Sledeće metoda provere identiteta su podržane samo kod TCP/IP soket konekcije:

***krb4***

* Kerberos V4 se koristi za autentifikaciju korisnika.

***krb5***

* Kerberos V5 se koristi za autentifikaciju korisnika.

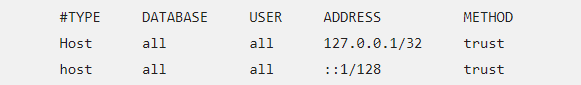
Slede neki tipični primeri konfigurisanja PostgreSQL autentifikacije:

Primer 1: Bilo koji korisnik može pristupiti bilo kojoj bazi podataka koristeći Unix soket, na klasteru PostgreSQL, prikazano je u primeru na slici 2. u nastavku:



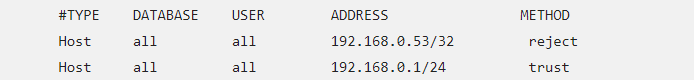
Slika 2., Izgled konfigurisane autentifikacije

Primer 2: Bilo koji korisnik na klasteru PostgreSQL može pristupiti bilo kojoj bazi podataka koristeći IP adresu localhost, kao što je prikazano na slici 3. u nastavku:



Slika 3., Izgled konfigurisane autentifikacije

Primer 3: Odbijaju se sve veze koje dolaze sa IP adrese 192.168.0.53, a veze koje potiču iz 192.168.0.1/24 se prihvataju, kao što je prikazano na slici 4. u nastavku:



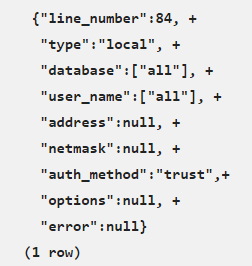
Slika 4., Izgled konfigurisane autentifikacije

PostgresSQL pruža veoma zgodan način za pregled pravila definisanih u datoteci pg\_hba.conf pružajući prikaz pod nazivom pg\_hba\_file\_rules na sledeći način:

Naredba za pregled pravila:

**postgres=# SELECT row\_to\_json(pg\_hba\_file\_rules, true) FROM pg\_hba\_file\_rules limit 1;**

Rezultat naredbe predstavlja naredna slika 5.:

****

Slika 5. Prikaz jednog pravila iz konfiguracionog fajla

### Najbolje prakse autentifikacije

Najbolje prakse autentifikacije zavise od kompletne infrastrukture, prirode aplikacije, karakteristike korisnika, osetljivosti podataka i tako dalje. Na primer, postavke u nastavku su najčešća praksa kod start-up kompanija: aplikacija baze podataka uključuje server baze podataka, hostuje se na istoj mašini i koristi se samo s jedne fizičke lokacije od strane korisnika unutar kompanije.

Često su serveri baze podataka izolovani od spoljašnjosti korišćenjem firewall-a, u tom slučaju može se koristiti SCRAM-SHA-256 metod za autentifikaciju i ograničenje IP adresa i tako će server baze podatak prihvatati adrese u određenom opsegu. Takođe je važno da se za povezivanje sa bazom ne koristi računar superusera, jer ako on bude hakovan, bio bi izložen čitav klaster baze podataka.

Ako server aplikacije tj. biznis logika i server baze podataka nisu na istoj mašin, vi možete koristiti jači metod autentifikacije LDAP i Kerberos. Međutim, za manje aplikacije gde server baze podataka i aplikacija na istoj mašini, SCRAM-SHA-256 metod i ograničenje adresa za pristup za localhost mogu biti dovoljne.

Za autantifikaciju preko aplikacije preporučuje se samo jedan korisnik i dobro bi bilo da se smanji maksimalan broj dozvoljenih veza da bi se prilagodili resursi PostgreSQL. Bilo bi dobro da postoji još jedan nivo sigurnosti u poslovnoj logici za razlikovanje različitih korisnika za prijavu. Za korisnike u realnom svetu, LDAP ili Kerberos autentifikacija je poželjna.

Ako se serveru baze podataka pristupa iz spoljnog sveta, korisno je šifrovati sesije pomoću SSL sertifikata kako bi se izbeglo njuškanje paketa.

Poželjno je i još zaštititi servere baza podataka koji veruju svim localhost vezama, jer svako ko pristupa localhost može pristupiti i serveru baze podataka.(2)

## Sistem uloga

Često se kod aplikacija koriste uloge za prijavu i konfigurisanje veza prema bazi podataka i alata za povezivanje. Potrebno je implementirati još jedan nivo sigurnosti da bi se osiguralo da je korisnik koji koristi aplikaciju ovlašćen za obavljanje određenog zadatka. Ova logika se najčešće primenjuje u poslovnoj logici aplikacija. U nastavku biće predstavljen koncept PostgreSQL uloga i biće pokazano kako se kreiraju korisničke ulige i grupne uloge.

PostgreSQL koristi koncept uloga za upravljanje dozvolama za pristup bazi podataka. Uloga može biti korisnik ili grupa, u zavisnosti od podešavanja uloga. Uloga koja sadrži login se zove naziva korisnik(user). Uloga može biti član drugih uloga, koje se nazivaju grupe.

Od verzije 8.1. PostgreSQL koristi koncept uloga za dodavanje koncepata korisnika i grupa. Za kreiranje nove uloge potrebno je koristiti CREATE ROLE komandu kao u nastavku:

**CREATE** **ROLE** role\_name;

Komandom u nastavku možete pribaviti sve dostupne uloge u klasteru:

**SELECT** rolname

**FROM** pg\_roles;

Atributi uloga

Atributi uloga baze podataka definišu privilegije uloge, uključujući prijavu, mogućnost stvaranja nove baze, kreiranje uloga, lozinke itd.

Naredba u nastavku stvara ulogu koja sadrži prijavu sa lozinkom i datum validnosti:

**CREATE** **ROLE** doe

**WITH** **PASSWORD** 'pgSecpas1970'

VALID **UNTIL** '2020-01-01';

Sledeća naredba stvara ulogu koja ima status superusera, koji može zaobići sve provere:

**CREATE** **ROLE** bigboss SUPERUSER;

Naravno da bi se kreirala uloga superuser neopgodno je da korisnik koji ga kreira takođe ima status superusera.

Ako je potrebno da uloga ima privilegiju za kreiranje baze podataka, potrebno je izvršiti sledeču naredbu:

**CREATE** **ROLE** **admin** CREATEDB;

Komanda u nastavku ce kreirati ulogu koja ima privilegiju kreiranja uloga:

**CREATE** **ROLE** **security** CREATEROLE;

Pripadnost ulogama

Lakse je upravljati ulogama kao grupama tako da možete dodeliti ili opozvati privilegije iz grupe kao celine. U PostgreSQL-u kreirate ulogu koja predstavlja grupu, a zatim dodeliti individualnim korisnicima članstvo u toj grupi uloga.

Po konvenciji, grupna uloga nema privilegiju LOGIN. Da bi se kreirala grupna uloga, može se koristiti naredba CREATE ROLE kao u nastavku:

**CREATE** **ROLE** group\_role;

Na primer naredna naredba kreira sales grupu uloga:

**CREATE** **ROLE** sales;

Nakon toga moguće je dodati korisničku ulogu u grupu uloga koristeći naredbu GRANT:

**GRANT** group\_role **to** user\_role;

Na primer da biste dodali ulogu korisnika doe u ulogu sales grupe potrebno je iskoristiti sledeću naredbu:

**GRANT** sales **TO** doe;

Da biste uklonili ulogu iz grupe uloga, potrebno je iskoristiti naredbu REVOKE:

**REVOKE** group\_role **FROM** user\_role;

Kao na primer ako želimo da uklonimo grupu uloga sales korisniku doe:

**REVOKE** sales **FROM** doe;

PostgreSQL ne dozvoljava da postoje kružne petlje u kojima je uloga član druge uloge i obrnuto.

### Nasleđivanje grupnih i korisničkih uloga

Korisnička uloga može koristiti grupne uloge na sledeće načine:

* Prvi način, korišćenjem SET ROLE naredbe korisnička uloga postaje grupna uloga, što znači da će korisnička uloga koristiti privilegije grupne uloge umesto originalnih privilegija korisničke uloge. Pored toga, bilo koji objekat kreiran u bazi u toj sesiji će biti dodat grupnoj ulozi, umesto da bude pridodat korisničkoj ulozi.
* Drugi način, kada korisnička uloga poseduje atribut INHERIT, automatski će imati privilegije grupnih uloga čiji je član, uključujući i sve privilegije nasleđene grupnom ulogom. Može se videti u nastavku:

**CREATE** **ROLE** doe LOGIN INHERIT;

**CREATE** **ROLE** sales NOINHERIT;

**CREATE** **ROLE** marketing NOINHERIT;

**GRANT** sales **TO** doe;

**GRANT** marketing **TO** sales;

Ako se na PostgreSQL neko poveže kao doe imaće privilegije koje sadrži uloga doe i plus dodatne privilegije koje obezbedjuje uloga sales, zato što uloga doe sadrži INHERIT atribut. Međutim, neće sadržati privilegije uloge marketing zato što je NOINHERIT atribut definisan za ulogu sales.

Nakon što se izvrši komanda:

**SET** **ROLE** sales;

Postojaće samo privilegije koje obezbedjuje sales uloga.

Da bi se vratile originalne privilegije, moguće je izvršiti sledeću naredbu:

**RESET** **ROLE**;

Jedino su privilegije vezane za objekte u bazi podataka nasledne. LOGIN, SUPERUSER, CREATEROLE i CREATEDB su posebne uloge koje se ne mogu naslediti kao obična privilegija.

### Uklanjanje uloga

Moguće je koristiti DROP ROLE naredbu za brisanje grupnih i korisničkih uloga.

**DROP** **ROLE** role\_name;

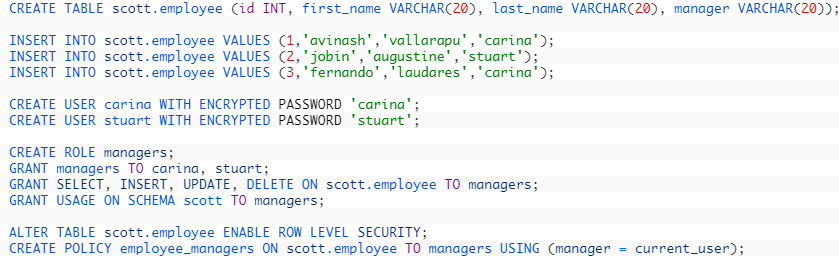
Pre uklanjanja uloge, morate ponovo dodeliti ili ukloniti sve objekte koje poseduje ili opozvati njihove privilegije.

Ako uklonite grupnu ulogu, PostgreSQL automatski opoziva sva članstva u grupi automatski. Na korisničke uloge ne može uticati.(4)

## Zaštita na nivou reda

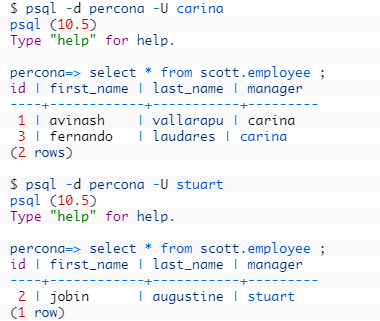
Počevši od verzije 9.5, PostgreSQL implementira zaštitu na nivou reda, što može ograničiti pristup samo određenog dela redova u tabeli za određenog korisnika. Obično korisnik ima kombinaciju privilegija SELECT, INSERT, DELETE i UPDATE što bi omogućavalo pristup svim redovima u tabeli. Međutim putem ograničenja na nivou reda, takve privilegije će biti ograničene na podskup redova za koje ce korisnik imati pristup.

U sledećem primeru na slici 6. će biti kreirana tabela zaposlenih (employee) i dva korisnika menadžera (carina i stuart). Zatim će se u tabeli omogućiti zaštita na nivou reda i kreiraće se pravila koja dozvoljavaju samo određenim menadžerima da gledaju i menjaju određene zapise:



Slika 6. Kreiranje tabela i primene zaštite na nivou reda

Nakon što smo izvršili prethodne komande i dodali zaštitu na nivou reda, proverićemo pristup za oba korisnika pomoću naredbi na slici 7.:



Slika 7. Izvršenje naredbi SELECT i rezultati kod različitih korisnika

U primeru vidimo da će svaki korisnik pristupiti onim kolonama za koje ima privilegije.(5)

## Zaključak

Sve u svemu, postoje mnogo načina da se osigura i obezbedi sistem, takođe postoji i više slojeva koji ga mogu obezbeđivati. Naravno sve zavisi od tipa aplikacije i načina njenog korišćenja. Svakako da će svaka dodatna metoda koštati malo više vremena, ali neovlašćeni pristup podacima, krađa podataka ili obaranje servera mogu mnogo ugroziti taj sistem, kompaniju i reputaciju kompanije. Tako da je veoma važno da ulozimo vreme i resurse shodno tipu naseg sistema i da sigurnost i bezbednost budu na visokoj poziciji u listi prioriteta.

## Literatura

1. Zvanična dokumentacija o sigurnosti kod PortgreSQL <https://www.postgresql.org/docs/7.0/security.htm>
2. Praksa vezana za autentifikaciju kod PostgreSQL <https://hub.packtpub.com/postgresql-security-a-quick-look-at-authentication-best-practices-tutorial/>
3. 10 saveta za sigurnost kod PostgreSQL

<https://www.upguard.com/articles/10-ways-to-bolster-postgresql-security>

1. PostgreSQL upravljanje ulogama <https://www.postgresqltutorial.com/postgresql-roles/>
2. Osigurati PostgreSQL okruženje <https://www.percona.com/blog/2018/09/21/securing-postgresql-as-an-enterprise-grade-environment/>
3. Saveti kako osigurati PostgreSQL bazu podataka <https://severalnines.com/database-blog/how-secure-your-postgresql-database-10-tips>
4. PostgreSQL dokumentacija autentifikacija sa lozinkom <https://www.postgresql.org/docs/11/auth-password.html>