UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET

Seminarski rad II

SIGURNOST KOD POSTGRESQL BAZE PODATAKA

Student:
Ivana Petković, br. ind. 1787

Mentor:
Doc. dr Aleksandar Stanimirović

Niš, maj 2024. godina

Sadržaj

1. Uvod
2. Sigurnost baza podataka
   - 2.1. Zbog čega je od tolikog značaja?
   - 2.2. Najčešći faktori narušavanja sigurnosti
   - 2.3. Mehanizmi zaštite
   - 2.4. Najbolje prakse
3. Sigurnost kod PostgreSQL baze podataka
   - 3.1. Autentifikacija klijenta
     - 3.1.1. pg_hba.conf fajl
     - 3.1.2. Mapiranje korisničkih imena
     - 3.1.3. Lightweight Directory Access Protocol(LDAP) metod autentifikacije
     - 3.1.4. SSL metod autentifikacije
   - 3.2. Korisničke uloge
     - 3.2.1. Superuser(Superkorisnik)
     - 3.2.2. Davanje ograničenih ovlašćenja superkorisnika određenim korisnicima
     - 3.2.3. Kreiranje novog korisnika
     - 3.2.4. Uklanjanje korisnika bez brisanja njegovih podataka
   - 3.3. Kontrola pristupa
     - 3.3.1. Dozvola pristupa nekog korisnika tabeli
     - 3.3.2. Dozvola pristupa korisnika određenim kolonama
     - 3.3.3. Dozvola pristupa korisnika određenim redovima
     - 3.3.4. Privremeno sprečavanje korisnika da se poveže
4. Zaključak
5. Literatura

Sigurnost kod PostgreSQL baze podataka

Autentifikacija klijenta

Autentifikacija je proces kojim server baze podataka uspostavlja identitet klijenta i određuje da li je klijentskoj apliakciji dozvoljeno da se poveže sa traženim korisničkim imenom baze podataka.

Korisničke uloge

Korisničke uloge definišu dozvole i privilegije korisnika.
Uloge mogu predstavljati individualne korisnike ili grupe korisnika.

Kontrola pristupa

Kontrola pristupa bazi podataka definiše ko i na koji način može pristupati podacima unutar baze podataka. Ovo uključuje različite dozvole i ograničenja korisnika i zavisi od privilegija koje imaju sami korisnici.

Metode autentifikacije

- Trust autentifikacija
- Password autentifikacija
- GSSAPI autentifikacija
- SSPI autentifikacija
- Provera identiteta
- Peer autentifikacija
- LDAP autentifikacija
- RADIUS autentifikacija
- Autentifikacija sertifikatom
- PAM autentifikacija
- BSD autentifikacija

pg_hba.conf fajl

```
local             database  user  auth-method [auth-options]
host              database  user  address     auth-method  [auth-options]
hostssl           database  user  address     auth-method  [auth-options]
hostnossl         database  user  address     auth-method  [auth-options]
hostgssenc        database  user  address     auth-method  [auth-options]
hostnogssenc      database  user  address     auth-method  [auth-options]
host              database  user  IP-address  IP-mask      auth-method  [auth-options]
hostssl           database  user  IP-address  IP-mask      auth-method  [auth-options]
hostnossl         database  user  IP-address  IP-mask      auth-method  [auth-options]
hostgssenc        database  user  IP-address  IP-mask      auth-method  [auth-options]
hostnogssenc      database  user  IP-address  IP-mask      auth-method  [auth-options]
include           file
include_if_exists file
include_dir       directory
```

Mapiranje korisničkih imena

- U nekim slučajevima, korisničko ime za autentifikaciju se razlikuje od PostgreSQL korisničkog imena. Na primer, ovo se može desiti kada se koristi eksterni sistem za autentifikaciju, kao što je provera autentičnosti sertifikata ili bilo koji drugi eksterni sistem. U ovim slučajevima možda ćete morati da omogućite eksterno autentifikovanom korisniku da se poveže kao više korisnika baze podataka. U takvim slučajevima možete odredti pravila za mapiranje spoljašnjeg korisničkog imena u odgovarajućeg korisnika(ulogu) baze podataka.

- Mape korisničkih imena su definisane u datoteci koja se podrazumevano zove **pg_ident.conf**(jer je prvobitno korišćena za metod provere identiteta) i čuva se u direktorijumu podataka klastera. Moguće je promeniti ime ove datoteke preko konfiguracionog parametra ident_file u postgresql.conf.

- Linije u datoteci mapa imaju format:
  ```
  map-name system-username database-username
  ```

- **PRIMER:**
  ```
  salesmap /^(.*)@sales\.comp\.com$ \1
  salesmap /^(.*)@sales\.comp\.com$ sales
  salesmap manager@sales.comp.com auditor
  ```

Lightweight Directory Access Protocol(LDAP) metod autentifikacije

- Ovaj metod autentifikacije funkcioniše slično kao i autentifikacija lozinkom, osim što koristi LDAP server kao metod provere lozinke. LDAP se koristi samo za validaciju parova korisničkog imena/lozinke. Stoga korisnik mora već postojati u bazi podataka pre nego što se LDAP može koristiti za autentifikaciju.
- LDAP autentifikacija može raditi u dva režima.
- U prvom režimu, koji se zove jednostavni režim povezivanja, server će se vezati za prepoznatljivo ime u formatu prefiks korisničko ime sufiks.
- U drugom režimu koji se zove režim pretrage i vezivanja(search+bind), server se prvo povezuje na LDAP direktorijum sa fiksnim korisničkim imenom i lozinkom, navedenim sa ldapibindnn i ldapbindpasswd, i vrši pretragu korisnika koji pokušava da se prijavi na bazu podataka.

- **PRIMER:**

```
host all all 10.10.0.1/16 ldap \
ldapserver=ldap.our.net ldapprefix="cn=" ldapsuffix=",
dc=our,dc=net"
```

SSL metod autentifikacije

- Ovaj metod podrazumeva korišćenje SSL klijentskih sertifikata za izvršavanje autentifikacije. Stoga je dostupan samo za SSL konekcije.
- Konfiguracija SSL-a se obavlja u okviru **postgresql.conf fajla**.
- Sledeće opcije konfiguracije su podržane za autentifikaciju SSL sertifikatom (konfiguracije dakle u okviru **pg_hba.conf** datoteke za određivanje pravila autentifikacije; za auth-method je izabrana opcija cert):
  map – omogućava mapiranje između sistemskih i korisničkih imena.
- Nema potrebe koristiti opciju clientcert sa cert autentifikacijom jer cert autentifikacija već podrazumeva clientcert=verify-full, što znači da je sertifikat klijenta u potpunosti verifikovan.

PRIMER 1

1) Generisanje sertifikata:

```
openssl genrsa 2048 > client.key
openssl req -new -x509 -key server.key -out client.crt
```

2) Konfiguracija servera:

```
hostssl all all 0.0.0.0/0 scram-sha-256 clientcert=1
```

3) Konfiguracija klijenta:

```
chmod 0600 ~/.postgresql/postgresql.key
```

PRIMER 2

```
host all all 10.10.10.10/32 scram-sha-256
hostssl all all 10.10.10.10/32 trust clientcert=1
hostssl all all all scram-sha-256 clientcert=1
```

Korisničke uloge

- PostgreSQL upravlja dozvolama za pristup bazi koristeći koncept uloga. Korisničke uloge definišu dozvole i privilegije korisnika. Uloga se može smatrati ili korisnikom baze podataka ili grupom korisnika, u zavisnosti od toga kako je postavljena. Uloge mogu posedovati objekte baze podataka (npr. tabele i funkcije) i mogu dodeliti privilegije nad tim objektima drugim ulogama. Štaviše, može se dodeliti članstvo u ulozi drugoj ulozi, čime se dozvoljava toj ulozi da koristi privilegije dodeljene drugoj ulozi. Koncept uloge obuhvata koncepte korisnika i grupa.

- Uloge baze podataka su konceptualno potpuno odvojene od korisnika operativnog sistema. Uloge baze podataka su globalne u celoj instalaciji klastera baze podataka(a ne po pojedinačnoj bazi podataka).

Korisničke uloge

Role	Allowed Access
pg_read_all_data	Read all data (tables, views, sequences), as if having SELECT rights on those objects, and USAGE rights on all schemas, even without having it explicitly. This role does not have the role attribute BYPASSRLS set. If RLS is being used, an administrator may wish to set BYPASSRLS on roles which this role is GRANTed to.
pg_write_all_data	Write all data (tables, views, sequences), as if having INSERT, UPDATE, and DELETE rights on those objects, and USAGE rights on all schemas, even without having it explicitly. This role does not have the role attribute BYPASSRLS set. If RLS is being used, an administrator may wish to set BYPASSRLS on roles which this role is GRANTed to.
pg_read_all_settings	Read all configuration variables, even those normally visible only to superusers.
pg_read_all_stats	Read all pg_stat_* views and use various statistics related extensions, even those normally visible only to superusers.
pg_stat_scan_tables	Execute monitoring functions that may take ACCESS SHARE locks on tables, potentially for a long time.
pg_monitor	Read/execute various monitoring views and functions. This role is a member of pg_read_all_settings, pg_read_all_stats and pg_stat_scan_tables.
pg_database_owner	None. Membership consists, implicitly, of the current database owner.
pg_signal_backend	Signal another backend to cancel a query or terminate its session.
pg_read_server_files	Allow reading files from any location the database can access on the server with COPY and other file-access functions.
pg_write_server_files	Allow writing to files in any location the database can access on the server with COPY and other file-access functions.
pg_execute_server_program	Allow executing programs on the database server as the user the database runs as with COPY and other functions which allow executing a server-side program.
pg_checkpoint	Allow executing the CHECKPOINT command.
pg_use_reserved_connections	Allow use of connection slots reserved via reserved_connections.
pg_create_subscription	Allow users with CREATE permission on the database to issue CREATE SUBSCRIPTION.

Superuser(Superkorisnik)

PostgreSQL super korisnik je korisnik koji zaobilazi sve provere dozvola, osim prava da se prijavi. To je opasna privilegija i ne bi trebalo da se koristi nesmotreno. Mnoge baze podataka u oblaku uopšte ne dozvoljavaju ovaj nivo privilegija. Normalno je postaviti stroge kontrole na korisnike ovog tipa.

Korisnik postaje superkorisnik kada je kreiran sa SUPERUSER skupom atributa:

```
CREATE USER username SUPERUSER;
```

Korisnik može biti lišen statusa super korisnika uklanjanjem atributa SUPERUSER pomoću ove komande:

```
ALTER USER username NOSUPERUSER;
```

Korisnik se kasnije može vratiti na status super korisnika pomoću komande:

```
ALTER USER username SUPERUSER;
```

Davanje ograničenih ovlašćenja superkorisnika određenim korisnicima

Uloga super korisnika ima neke privilegije koje se mogu zasebno dodeliti i ulogama koje nisu superkorisnici.

PRIMER 1:

```
pguser@hvost:~$ psql -U postgres
test2
...
test2=# create table lines(line text);
CREATE TABLE
test2=# copy lines from '/home/bob/names.txt';
COPY 37
test2=# GRANT ALL ON TABLE lines TO bob;
test2=# SET ROLE to bob;
SET
test2=> copy lines from '/home/bob/names.txt';
ERROR: must be superuser or have privileges of the pg_read_server_files
role to COPY from a file
HINT: Anyone can COPY to stdout or from stdin. psql's \copy command also
works for anyone.

test2=# GRANT pg_read_server_files TO bob;
```

Davanje ograničenih ovlašćenja superkorisnika određenim korisnicima

PRIMER 2:
Dodeljivanje privilegija za pravljenje rezervnih kopija korisniku

Sledećom komandom moguće je napraviti „rezervnog“ korisnika:

```
CREATE USER backup_user;
```

Za logičke rezervne kopije dodeliti mu ulogu pg_read_all_data:

```
GRANT pg_read_all_data TO backup_user;
```

I atribut REPLICATION za pravljenje fizičkih rezervnih kopija:

```
ALTER USER backup_user WITH REPLICATION;
```

Kreiranje novog korisnika

Iz komandne linije se može primeniti komanda createuser:

```
pguser@hvost:~$ createuser bob
```

```
pguser@hvost:~$ createuser --interactive alice
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) y
Shall the new role be allowed to create more new roles? (y/n) n
```

Preko SQL upita:

```
CREATE USER bob;
CREATE USER alice CREATEDB;
```

Uklanjanje korisnika bez brisanja njegovih podataka

Kada pokušavamo da uklonimo korisnika koji poseduje neke tabele ili druge objekte baze podataka, dobijamo sledeću grešku koja onemogućava da korisnik bude izbačen:

```
testdb=# drop user bob;
ERROR: role "bob" cannot be dropped because some objects depend on it
DETAIL: owner of table bobstable
owner of sequence bobstable_id_seq
```

Da biste izmenili korisnike, morate da budete super korisnik ili da imate privilegiju CREATEROLE.

REŠENJE 1:

```
 pguser=# alter user bob nologin;
ALTER ROLE
```

REŠENJE 2:

```
pguser=# GRANT bob TO bobs_replacement;
GRANT
```

Kontrola pristupa

- Kontrola pristupa bazi podataka definiše ko i na koji način može pristupati podacima unutar baze podataka i od izuzetnog je značaja za sigurnost same baze podataka. Ovo uključuje različite dozvole i ograničenja korisnika i zavisi od privilegija koje imaju sami korisnici.

- **Primeri:**

1) Dozvola pristupa nekog korisnika tabeli
2) Dozvola pristupa korisnika određenim kolonama
3) Dozvola pristupa korisnika određenim redovima
4) Privremeno sprečavanje korisnika da se poveže

Dozvola pristupa nekog korisnika tabeli

```
CREATE GROUP webreaders;
CREATE USER tim;
CREATE USER bob;
REVOKE ALL ON SCHEMA someschema FROM PUBLIC;

GRANT USAGE ON SCHEMA someschema TO webreaders;

GRANT SELECT ON someschema.pages TO webreaders;
GRANT INSERT ON someschema.viewlog TO webreaders;
GRANT webreaders TO tim, bob;

GRANT INSERT, UPDATE, DELETE ON someschema.comments TO webreaders;

GRANT SELECT ON ALL TABLES IN SCHEMA someschema TO bob;
```

Dozvola pristupa korisnika određenim kolonama

```
CREATE TABLE someschema.sometable2(col1 int, col2 text);

GRANT SELECT, INSERT ON someschema.sometable2
TO somerole;
GRANT UPDATE (col2) ON someschema.sometable2
TO somerole;

SET ROLE TO somerole;
INSERT INTO someschema.sometable2 VALUES (1, 'One’);
SELECT * FROM someschema.sometable2 WHERE col1 = 1;

UPDATE someschema.sometable2 SET col2 = 'The number one’;
```

Što vraća izlaz:

```
UPDATE 1
```

```
UPDATE someschema.sometable2 SET col1 = 2;
```

Što će vratiti izlaz:

```
ERROR: permission denied for relation sometable2
```

Dozvola pristupa korisnika određenim redovima

PostgreSQL podržava davanje privilegija podskupu redova u tabeli pomoću RLS-a (RSL – Row Level Security).

PRIMER 1:

```
CREATE TABLE someschema.sometable3(col1 int, col2 text);
```

RLS mora biti omogućen na toj tabeli:

```
ALTER TABLE someschema.sometable3 ENABLE ROW LEVEL SECURITY;
```

```
GRANT SELECT ON someschema.sometable3 TO somerole;
```

```
SELECT * FROM someschema.sometable3;
col1 | col2
------+-----------
1 | One
-1 | Minus one
(2 rows)
```

Dozvola pristupa korisnika određenim redovima

```
CREATE POLICY example1 ON someschema.sometable3
FOR SELECT
TO somerole
USING (col1 > 0);
```

```
SELECT * FROM someschema.sometable3;
col1 | col2
------+-----------
1 | One
(1 row)
```

Dozvola pristupa korisnika određenim redovima

PRIMER 2:

```
CREATE POLICY example2 ON someschema.sometable3
FOR INSERT
TO somerole
WITH CHECK (col1 > 0);


GRANT INSERT ON someschema.sometable3 TO somerole;
SELECT * FROM someschema.sometable3;
col1 | col2
------+-----------
1 | One
(1 row)

INSERT INTO someschema.sometable3 VALUES (2, 'Two');
SELECT * FROM someschema.sometable3;
col1 | col2
------+-----------
1 | One
2 | Two
(2 rows)
```

Privremeno sprečavanje korisnika da se poveže

Ovom komandom možete privremeno sprečiti korisnika da se prijavi:

```
pguser=# alter user bob nologin;
ALTER ROLE
```

A ovom možete dozvoliti korisniku da se ponovo prijavi:

```
pguser=# alter user bob login;
ALTER RPLE
```

Ograničavanje broja konkurentnih konekcija korisnika

```
pguser=# alter user bob connection limit 0;
ALTER ROLE

pguser=# alter user bob connection limit 10;
ALTER ROLE

pguser=# alter user bob connection limit -1;
ALTER ROLE
```

Hvala na pažnji!