POSTGRESQL DATUBĀZES DROŠĪBAS KONFIGURĒŠANAS KONTOLSARAKSTS
POSTGRESQL DATABASES SECURITY CONFIGURATION CHECKLIST

Autors: **Rinalds GUDRIKS**, e-pasts: xrinbo@gmail.com
Zinātniskā darba vadītājs: **Doc. Dr.sc.ing.**, **Sergejs KODORS,** e-pasts: sergejs.kodors@rta.lv
Rēzeknes Tehnoloģiju akadēmija
Atbrīvošanas aleja 115, Rēzekne

Abstract. *The paper provides PostgreSQL configuration checklist to make databases safer. The main part describes with examples about vulnerabilities and how to solve them.*
Keywords: *configuration checklist, databases, PostgreSQL, security, vulnerabilities.*

Ievads

PostgreSQL ir jaudīga, atvērtā koda objektu relācijas datubāzes pārvaldības sistēma (DBVS), kas izmanto un papildina *SQL* valodu kopā ar citām funkcijām, droši uzglabā datus un spēj tikt galā ar sarežģītām datu slodzēm. Atbalsta daudzas funkcijas un iespējas no *SQL2003* standarta (*ISO/ IEC 9075*). Šobrīd jaunākā versija *PostgreSQL* ir *11*, tā ir izlaista 2018. gada oktobrī un atbalsta vismaz 160 no 179 *SQL* standartiem. [1]

PostgreSQL ir ieguvusi spēcīgu reputāciju par pierādīto arhitektūru, uzticamību, datu integritāti, robustu funkciju kopumu un atvērtā pirmkoda kopienas atbalstu, kas nodrošina novatorisku risinājumus. [1]

DBVS mērķis ir palīdzēt izstrādātājiem veidot lietojumprogrammas, administratoriem aizsargāt datu integritāti un veidot kļūdu-noturīgu vidi, kā arī palīdzēt pārvaldīt datus neatkarīgi no to apjoma. Papildus tam, ka *PostgreSQL* ir bezmaksas un atvērta pirmkoda, šī DBVS ir arī paplašināma. Tas nozīmē, ka lietotājs var definēt savus datu tipus, veidot pielāgojamas funkcijas, pat rakstīt kodu dažādās programmēšanas valodās nepārkompilējot datubāzi.[2]

Tikko instalēts *PostgreSQL* satur standarta konfigurāciju, lai lietotājs varētu izmantot DBVS. Jāatzīst, ka standarta konfigurācija nesatur sevī visus iespējamos drošības parametrus, tomēr lietotājs pats var veikt konfigurāciju un paaugstināt savu datu drošību.

Pētījuma mērķis ir izstrādāt *PostgreSQL* datubāzes drošības konfigurēšanas kontrolsarakstu balstoties uz ekspertu rekomendācijām un autora pieredzi.

Pētījuma objekti un metodes

PostgreSQL drošības konfigurēšanas kontrolsaraksts tiks izstrādāts balstoties uz ekspertu rekomendācijām rakstos *"How to Achieve PostgreSQL Security"* [3], *"OWASP Backend Security Project PostgreSQL Hardening"* [4], *"Security Best Practices for Postgres"* [5] un *PostgreSQL* dokumentacijas.

Ņemot vērā autora pieredzi informācijas tehnoloģiju nozarē, kā arī dažādu DBVS izmantošanā, tai skaitā arī *PostgreSQL*, autors sniedz arī savu viedokli par izstrādātā drošības konfigurēšanas kontrolsaraksta būtiskumu.

Rezultāti un to izvērtējums

Tiklīdz ir instalēts svaigs *PostgreSQL* datubāžu serveris ir nepieciešams to nokonfigurēt tā, lai tas būtu drošs pirms to sākt izmantot produkcijā. Tiks aprakstīts kontrolsaraksts ar drošības konfigurācijām, kuras tiek rekomendētas:

- lietotāju autentifikācijas kontrole;
- servera konfigurācija;
- lietotāju un lomu pārvaldība;
- superlietotāju pārvaldība;

http://dx.doi.org/10.17770/het2019.23.4386

- datu šifrēšana (izmantojot SSL);
- datu šifrēšana;
- žurnāls;
- atjauninājumi;
- rindu līmeņa aizsardzība.

Tālākā tekstā katrs punkts tiks aprakstīts sīkāk un dotas rekomendācijas kā uzlabot drošību.

Lietotāju autentifikācijas kontrole

Kad tiek instalēts *PostgreSQL*, tad fails ar nosaukumu *pg_hba.conf* tiek izveidots direktorijā. Šis fails kontrolē lietotāju autentifikāciju. Oficiālajā *PostgreSQL* dokumentācijā ir teikts, ka katra rinda šajā failā apraksta pieslēgšanās metodi, lietotāja IP adreses (no kurienes tiks veikts savienojums), datubāzes nosaukumu, lietotājvārdu un autorizācijas metodi (skat. 1. att., piemēru, kā izskatās konfigurācijas fails, skat. 2. att.). [3]

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
```

1.att. ***PostgreSQL* konfigurācijas faila parametri**

```
# Allow any user from any host with IP address 192.168.93.x to connect
# to database "postgres" as the same user name that ident reports for
# the connection (typically the operating system user name).
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
 host     postgres             all             192.168.93.0/24         ident
# Reject any user from any host with IP address 192.168.94.x to connect
# to database "postgres
# TYPE  DATABASE        USER            ADDRESS                 METHOD
 host     postgres             all             192.168.94.0/24         reject
```

2.att. ***pg_hba.conf* faila piemērs**

Lai būtu pārliecība, ka savienojums būs drošs, tad nekad netraucēs pievienot attiecīgu rindu konfigurācijas faila beigās (skat. 3. att.) [3]

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
 host     all              all             0.0.0.0/0          reject
```

3. att. ***pg_hba.conf* faila piemērs, kad pieslēgšanās no jebkuras neaprakstītas IP adreses tiek liegta viesiem lietotājiem uz visām DB**

Servera konfigurācija

Līdz ar *pg_hba.conf* failu tiek izveidots arī *postgresql.conf* fails, kas ir paredzēts *PostgreSQL* servera konfigurācijai. Ir iespēja rediģēt šo failu, lai uzlabotu drošību. Lai to izdarītu ir jāatrod parametrs *listen_address*, un jānorāda *IP* adreses vai tīkls, kas drīkst pieslēgties pie *PostgreSQL* servera. Jāizvairās no tādām vērtībām kā "*", "0.0.0.0", "::", kas norāda uz to, ka pie servera var slēgties no jebkuras *IP* adreses.

Papildus drošībai var arī nomainīt noklusējuma portu (pēc noklusējuma tas ir *5432*). To var izdarīt nomainot vērtību parametram *port*.

Tādus parametrus kā *work_mem, maintenance_work_mem, temp_buffer, temp_file_limit, max_prepared_transactions* arī ir jāņem vērā, lai pasargātos no uzbrukumiem. Šiem parametriem var piešķirt dažādus līmeņus (db, user, session), lai samazinātu ietekmi no uzbrukumiem. [3-6]

Lietotāju un lomu pārvaldība

Zelta likums lietotāju pārvaldībā ir dot lietotājam tik daudz tiesību, cik viņiem ir nepieciešams. Pārvaldība ne vienmēr ir viegli īstenojama, bet ja to neīstenot sākumā, tad beigās var būt daudz sarežģītāk to ieviest. *PostgreSQL* tas ir īstenots izmantojot lomas, šajā gadījumā var izveidot trīs lomas:

- *role role* (aprakstīta ar *r_*)
- *group role* (aprakstīta ar *g_*)
- *user role* (lietotājvārds)

Loma *r_* būs tā, kas apraksta kādas tiesības būs objektam. Loma *g_* apraksta kādas lomas tiks pievienotas (lomu kopums). *User* loma tiks pievienota ar vienu vai vairākām *g_* lomām, kā arī dos *login* (autorizācijas) tiesības.

Zemāk ir piemērs kā var izmantot lomas. Tiks izveidota *read only* (tikai apskatīt) loma priekš *example_schema* un šī loma tiks pievienota attiecīgam lietotājam:

Tiek izveidota *read only* loma un pievienota pie objekta (skat. 4. att.)[3][5]

```
CREATE ROLE r_example_ro NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE NOREPLICATION;
GRANT USAGE ON SCHEMA example to r_example_ro;
GRANT SELECT ON ALL TABLES IN SCHEMA example to r_example_ro;
ALTER DEFAULT PRIVILEGES IN SCHEMA example GRANT SELECT ON TABLES TO r_example_ro;
```

4. att. ***read only*** **lomas pievienošanas piemērs**

Tiek izveidota *read only* lomu grupa un tai piešķir *r_example_ro* lomu (skat. 5. att.) [3]

```
CREATE ROLE g_example_ro NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE NOREPLICATION';
GRANT r_example_ro to g_example_ro;
```

5. att. ***read only*** **lomu grupas pievienošanas piemērs**

Tiek izveidota lietotāja loma ar nosaukumu *app_user* un pievienota *read only* grupai (skat 6. att.). [3]

```
CREATE ROLE app_user WITH LOGIN ;
ALTER ROLE app_user WITH PASSWORD 'somePassword' ;
ALTER ROLE app_user VALID UNTIL 'infinity' ;
GRANT g_example_ro TO app_user;
```

6. att. **Lietotāja izveide un pievienošana grupai**

Šādā veidā var veidot lomas katrai lietotāju grupai, kas strādās ar datubāzi. Galvenais ir nemainīt tiesības katram lietotājam atsevišķi, bet to darīt izmantojot lomas (skat. 7. att.).[3]

```
REVOKE CONNECT ON my_database FROM PUBLIC;
GRANT CONNECT ON my_database TO r_example_ro;
```

7. att. **Piemērs kā pievienot papildus *CONNECT* tiesības grupai**

Vēlams arī ierobežot *SUPERUSER* (superlietotāja) pieeju, atļaut veikt pieslēgšanos tikai no *localhost* (lokālā servera). Kā arī ieteicams katram nolūkam izmantot savu lietotāju – aplikācijai savu lietotāju, rezerves kopijai savu lietotāju un katram lietotājam aprakstīt attiecīgus pieslēgšanās parametrus.[3-5]

Superlietotāju pārvaldība

Drošu paroļu politikas izmantošana ir viens no veidiem kā saglabāt datubāzes drošību un pasargātu to no uzlaušanas. Droša paroļu politika ir izmantot speciālus simbolus, ciparus, lielos un mazos burtus, parolei ir jāsatur vismaz 10 simbolus.

Ir iespēja arī izmantot autentifikācijas rīkus, tādus kā *LDAP* vai *PAM*, kas nodrošina paroles termiņa beigšanos un atkārtotu politikas izmantošanu, tāpat arī paredz lietotāju slēgšanu un autentifikācijas kļūdu pārvaldību.[3-4]

Datu šifrēšana (izmantojot SSL)

PostgreSQL atbalsta *SSL* izmantošanu savienojumu veidošanā, lai šifrētu klientu/servera komunikāciju un uzlabotu drošību. *SSL* (no angļu valodas *Secure Sockets Layer*) ir drošības tehnoloģijas standarts, kas izveido šifrētu savienojumu starp web serveri un pārlūkprogrammu. Tas paredz, ka visi dati, kas tiek pārsūtīti starp web serveri un pārlūkprogrammu būs privāti. Tā kā *PostgreSQL* klients pēc noklusējuma sūta vaicājumus vienkāršā tekstā un dati tiek nosūtīti nešifrēti, tad tā ir ievainojamība no tīkla *spoofinaga*.

Lai ieslēgtu *SSL*, postgresql.conf failā jāiestata parametrs *ssl*. Serveris pieņems gan parastus, gan *SSL* savienojumus tajā pašā portā, pēc noklusējuma kādu savienojumu izmantot ir klienta opcija, bet ir arī iespēja no servera puses iestatīt, lai katram vai arī tikai dažiem savienojumiem tiktu pieprasīta *SSL* izmantošana aprakstot to *pg_hba.conf* failā.[3-5][7]

Datu šifrēšana

Ir divi galvenie šifrēšanas veidi – vienā virzienā vai divus. Pirmajā veidā nav jārūpējas par datu atšifrēšanu lasāmā formā. Parasti tāda veida šifrēšanu izmanto paroļu šifrēšanā. Divkāršā šifrēšana paredz gan datu šifrēšanu, gan atšifrēšanu lasāmā formā. Šajā kategorijā ietilpst tādi dati kā kredītkartes un *SSN*.

Vienvirziena šifrēšanai *PostgreSQL* piedāvā paketi *pgcrypto*, kas paredz augstāka līmeņa drošību par *MD5,* jo izmantojot *MD5* var saprast, kuriem lietotājiem ir vienādas paroles, jo netiek izmantots *salt* (papildus *hash* priekš kriptēšanas), līdz ar ko visiem lietotājiem ar vienādām parolēm būs tādā pašā veidā *MD5* kodēta parole. Izmantojot *pgcrypto* paroles būs dažādas.

Parasti apskatot šifrētus datus nevar saprast kāds ir to saturs, tomēr *PostgreSQL* dokumentācijā ir aprakstīts arī kā autorizētiem lietotājiem dot iespēju atšifrēt datus.[3][8-9]

Žurnāls

PostgreSQL piedāvā vairākus variantus kā kontrolēt kas, ko, kad un kā ir darījis. Konfigurācijas failā *postgresql.conf* var norādīt cik daudz informācijas ir jāsatur žurnālam, piemēram, savienojuma izveidošana un atslēgšanās, vaicājumi, pagaidu failu lielums utt.[3]

Atjauninājumi

PostgreSQL regulāri tiek atjaunots, dažreiz ar funkcionālajiem atjauninājumiem nāk klāt arī drošības atjauninājumi, tāpēc ir ieteicams sekot līdzi *PostgreSQL* aizsardzības informācijas lapai. Jāņem vērā arī, ka operētājsistēmas ievainojamība var radīt datubāzes datu noplūdumus, tāpēc jāseko līdzi arī operētājsistēmas atjauninājumiem. [3-5]

Rindu līmeņa aizsardzība

Papildus *SQL* noklusējuma tiesību sistēmai, kas pieejama, izmantojos *GRANT*, tabulās var mainīt arī rindu drošības politiku (*Row-Level security*), kas balstoties uz lietotāju, ierobežo, kuras rindas var atgriezt, ievietot, atjaunināt vai dzēst vaicājumos ar datu modifikācijas komandām. Zemāk ir piemērs, kā izveidot politiku priekš lietotāja, lai atļautu tikai vadības līmeņa lomām piekļūt tikai viņu rindām (skat. 8. att.).[3][10]

```
CREATE TABLE accounts (manager text, company text, contact_email text);
ALTER TABLE accounts ENABLE ROW LEVEL SECURITY;
CREATE POLICY account_managers ON accounts TO managers USING (manager = current_user);
```

8. att. **Vadības līmeņa lietotājam atļaut apskatīt tikai viņa rindām**

Secinājumi

Šajā rakstā tika izstrādāts *PostgreSQL* drošības konfigurēšanas kontrolsaraksts balstoties ekspertu rekomendācijām, praktiskās pieredzes un literatūras analīzes.

Tika izstrādāti 8 punkti balstoties uz kuriem savienojums ar *PostgreSQL* datubāzēm būs drošāks, kas samazinās ievainojamību un uzlaušanas risku. Lietotāju pārvaldība būs daudz strukturētāka un konfigurējamāka. Datu plūsma aizsargātāka un darbības ar datubāzi varēs atsekot.

Savā praktikā autors apkopoja ekspertu rekomendācijas, lai samazinātu uzbrukumu skaitu un palielinātu datu drošību *PostgreSQL* datubāzēm un augstāk minētie punkti varbūt nespēs 100% aizsargāt datus un novērst to noplūdi, bet piespiedīs nelabvēļus papūlēties.

Summary

Once installation process of PostgreSQL database server is finished, it is necessary to protect it before going into production. This article provides PostgreSQL security configuration checklist based on expert advices, practical experience, and literature analysis.

Eight checkpoints were developed based on which the connection to the PostgreSQL databases will be much safer, risk of attacks will be lower, and vulnerability reduced. User management will be much more structured and configurable. The data flow will be more secure, and the database operations could be traced.

Author tried to find different solutions to reduce the number of attacks and to increase data security for PostgreSQL databases and the above checkpoints may not be able to protect and prevent leakage of 100%, but will force attackers to make greater effort.

Literatūras un avotu saraksts

1. What is PostgreSQL? [Tiešsaiste]
   Pieejams: https://www.postgresql.org/about/ [Piekļuve: 15.04.2019.]
2. Why use PostgreSQL? [Tiešsaiste]
   Pieejams: https://www.postgresql.org/about/ [Piekļuve: 15.04.2019.]
3. How to Secure PostgreSQL Database [Tiešsaiste]
   Pieejams: https://severalnines.com/blog/how-secure-your-postgresql-database-10-tips [Piekļuve: 15.04.2019.]
4. OWASP Backend Security Project PostgreSQL Hardening [Tiešsaiste]
   Pieejams: https://www.owasp.org/index.php/OWASP_Backend_Security_Project_PostgreSQL_Hardening [Piekļuve: 15.04.2019.]
5. Security Best Practises for Postgres [Tiešsaiste]
   Pieejams: https://info.enterprisedb.com/rs/069-ALB-339/images/security-best-practices-for-postgres.pdf?_ga=2.214934679.1028117103.1555331894-483784908.1552467399 [Piekļuve: 15.04.2019.]
6. The pg_hba.conf File [Tiešsaiste]
   Pieejams: https://www.postgresql.org/docs/9.6/auth-pg-hba-conf.html [Piekļuve: 15.04.2019.]
7. Secure TCP/IP Connection with SSL [Tiešsaiste]
   Pieejams: https://www.postgresql.org/docs/9.6/ssl-tcp.html [Piekļuve: 15.04.2019.]
8. pgcrypto [Tiešsaiste]
   Pieejams: https://www.postgresql.org/docs/current/pgcrypto.html [Piekļuve: 15.04.2019.]
9. Encription data with pgcrypto [Tiešsaiste]
   Pieejams: http://www.postgresonline.com/journal/archives/165-Encrypting-data-with-pgcrypto.html [Piekļuve: 15.04.2019.]
10. Row Security Policies [Tiešsaiste]
    Pieejams: https://www.postgresql.org/docs/9.6/ddl-rowsecurity.html [Piekļuve: 15.04.2019.]