Various Attacks and its Possible solutions to Secure data in PostgreSQL DBMS

By

Premnath Jangam

Ramanan Rajangam

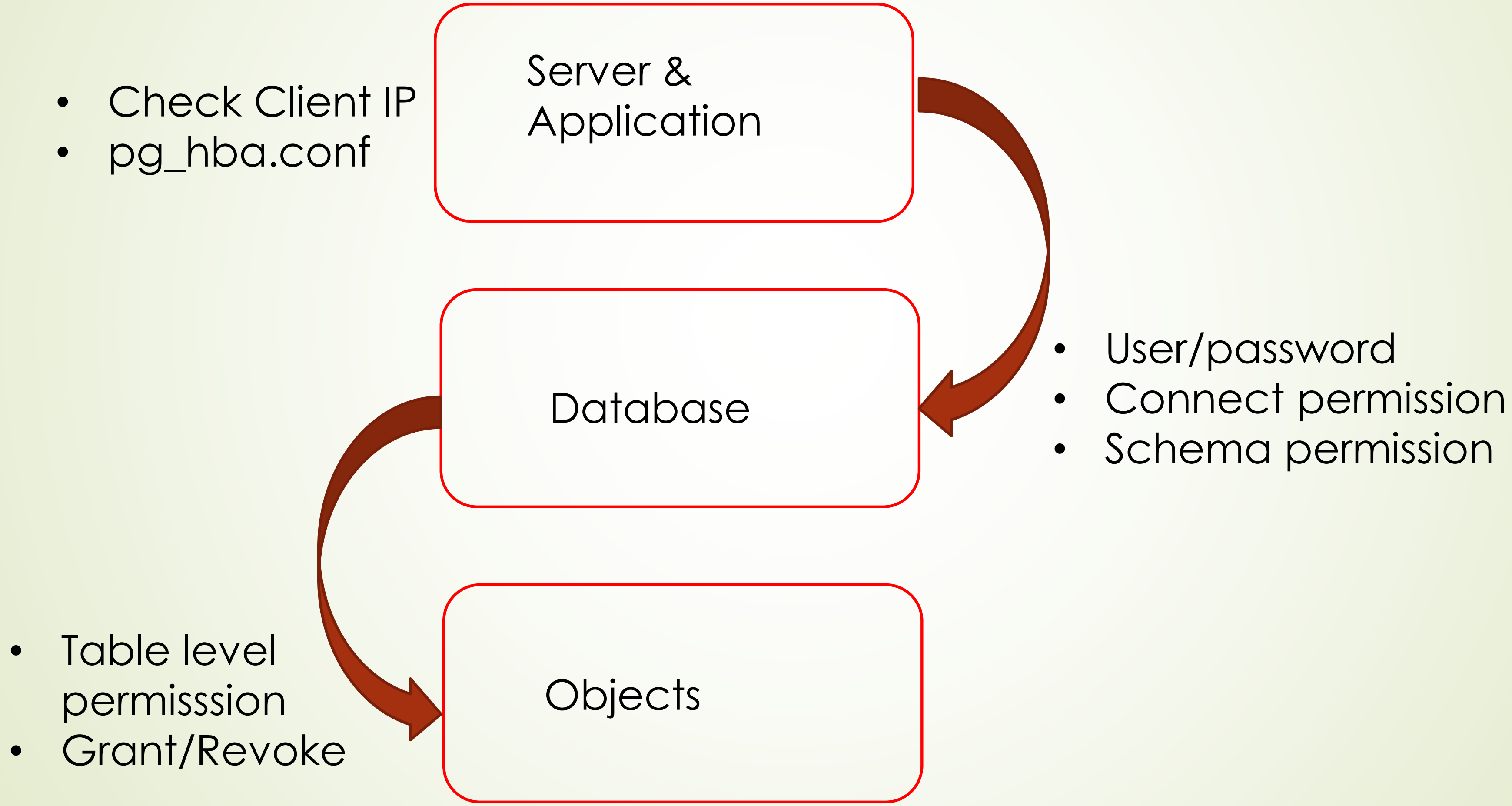
Abhinav Sagar

What is Database Security ? Is it Important ?

Top Most Popular Database attacks

- **Brute Force** (or not) cracking of weak or default usernames/passwords.
- Privilege Escalation
- Exploiting unused and unnecessary database services and functionality
- Targeting unpatched database vulnerabilities
- Stolen backup (unencrypted) tapes
- SQL injection

Levels of Security



- Check Client IP
- pg_hba.conf

Server & Application

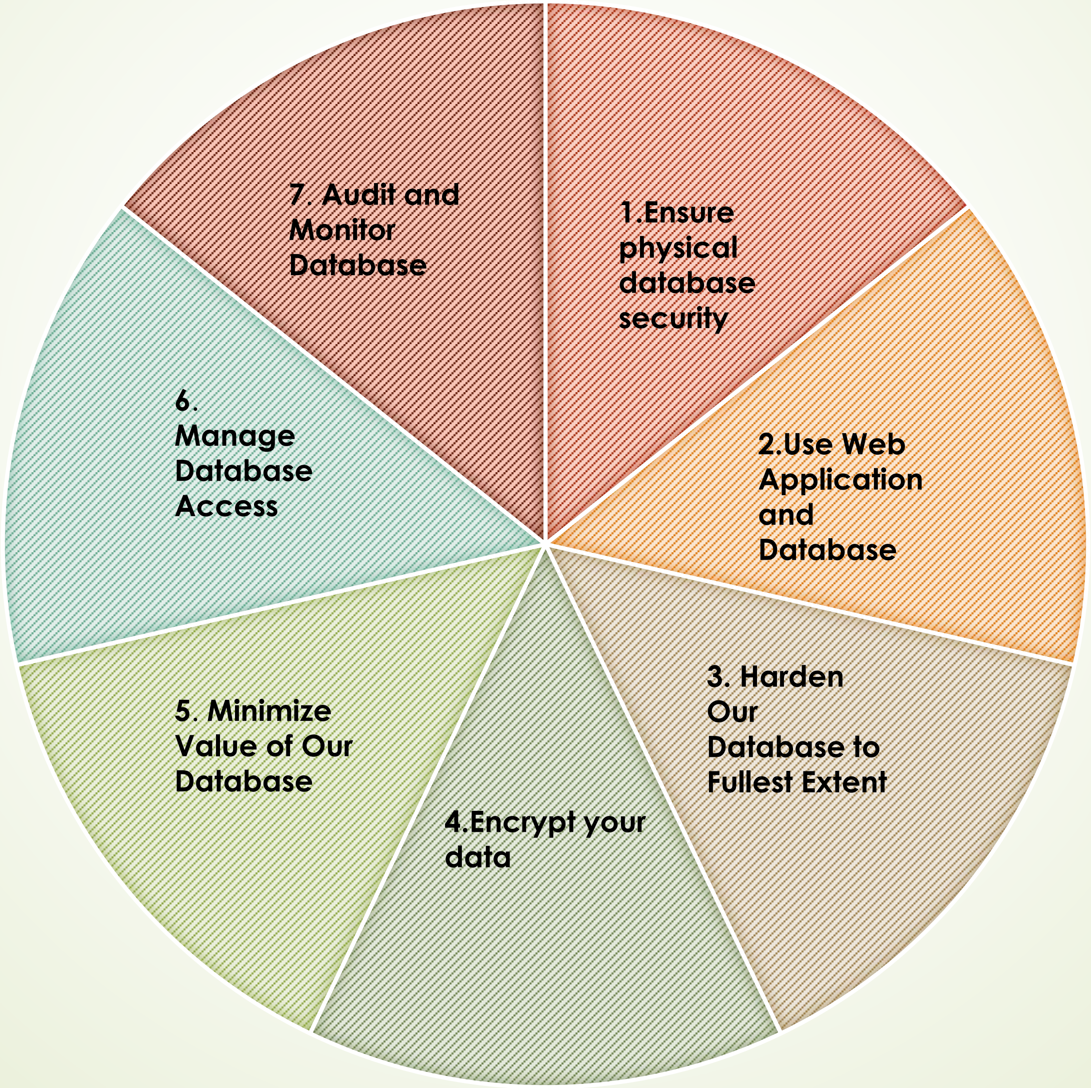
- User/password
- Connect permission
- Schema permission

Database

- Table level permisssion
- Grant/Revoke

Objects

Seven database security best practices



1.Ensure physical database security

2.Use Web Application and Database

3. Harden Our Database to Fullest Extent

4.Encrypt your data

5. Minimize Value of Our Database

6. Manage Database Access

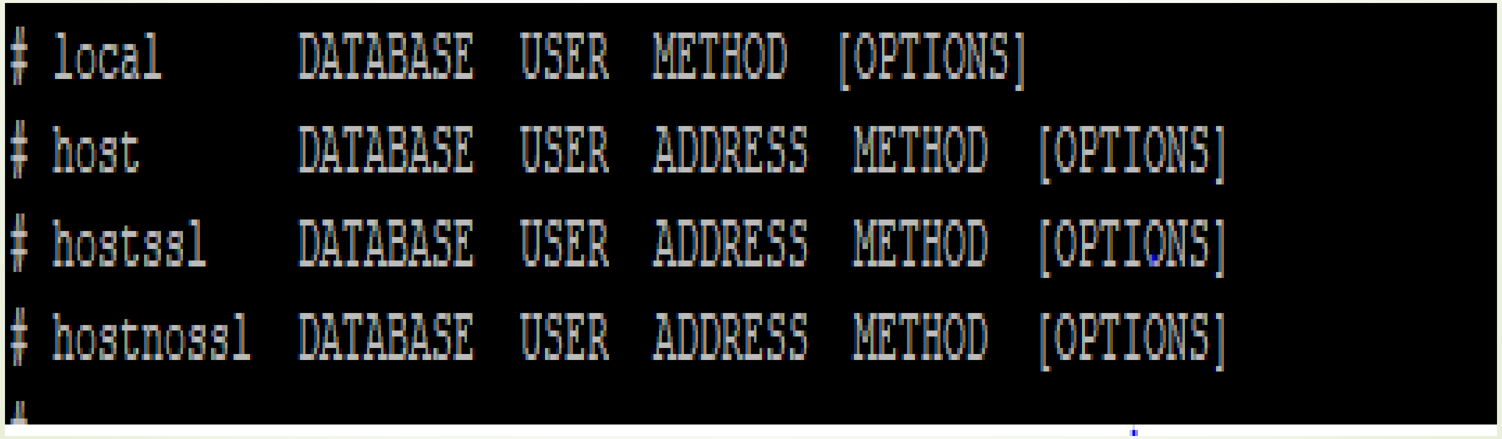
7. Audit and Monitor Database

Database Server Security

- The physical machine hosting a database should be housed in a secured, locked and monitored environment to prevent unauthorized entry, access of theft.
- App or Web servers should not be hosted on the same machine as the database server.
- Make sure proper firewalls are configured between app/web server & Database server.
- Disable public network access to database servers.
- Plan to have secondary server in geolocation for disaster recovery.
- Limit number of users to access the physical host

Authentication -   pg_hba.conf (Host access control)

- Host based access control files.
- Read at startup, any change requires reload.
- Each record specify connection type, database name, user name, client IP and method of authentication.
- Hostnames, IPv6 and IPv4 supported
- Various forms of pg_hba.conf

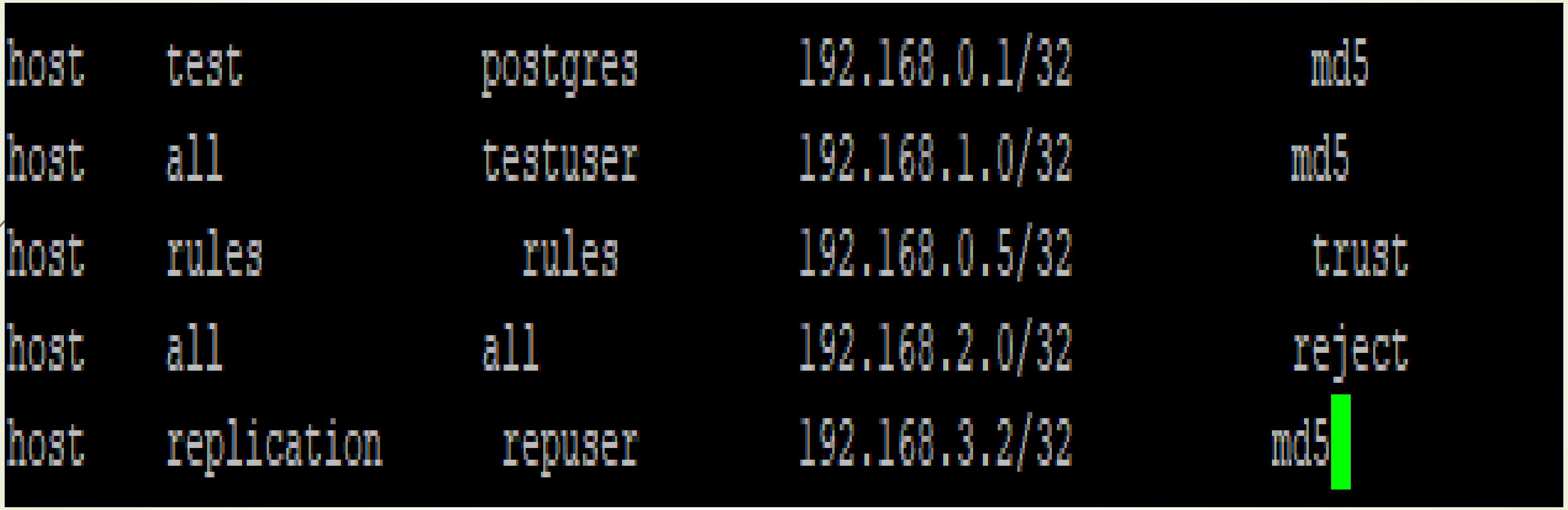


```
# local      DATABASE  USER  METHOD  [OPTIONS]
# host       DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
# hostssl    DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
# hostnossl  DATABASE  USER  ADDRESS  METHOD  [OPTIONS]
#
```

Auth Methods in pg_hba.conf :

- Trust
- Reject
- Scram-sha-256
- Md5
- Password
- Gss
- Sspi
- ident
- Peer
- Ldap
- Radius
- Cert
- Pam
- bsd

Examples – pg_hba.conf (auth methods)



```
host    test            postgres        192.168.0.1/32          md5
host    all             testuser        192.168.1.0/32         md5
host    rules             rules         192.168.0.5/32          trust
host    all             all             192.168.2.0/32         reject
host    replication      repuser        192.168.3.2/32        md5
```

Authorization (User Access)

- Use different users for different purpose
- A separate user for owning application database and schema
- Allow DBA’s to use their own user accounts
- Use a different (Non – superuser) for taking backups
- Allow replication connection using specific user from specific hosts
- Restrict superuser access
  - Allow Super user to make connection only from local host linux domain

Never use the Defaults

- Don’t use the default port of postgres cluster
- By Default each new DB has connect privilege granted to public schema

  revoke connect on <database> from public;

  grant connect on <database> to <username>;
- Use listen_address to control where our database is listening for connection
  - Make sure we don’t listen on public n/w interface
- Control the users who can connect from where
  - Use pg_hba.conf to control which user can connect to which specific database and from specific IP
  - Avoid using general rule like Database ALL user ALL or ip range ‘’0.0.0.0/0”

Auditing and Monitoring database

Database auditing allows administrators to track and analyze database activities in support of complex auditing requirements.

Mostly recommended log for audit

- Log connections
- DDL & DML changes
- Data changes
- Data views

Review your audit logs frequently for anomalous behavior

Minimize Value of Our Database

- Do not store any confidential data
- Retain data for compliance or other purposes

Avoid vulnerabilities - Timely patching

- Have a tab on various vulnerabilities announced by various companies
- Keep the OS and Database patched up to date

Encryption Levels

We can perform encryption on various levels

- Password storage encryption
- Encryption for specific columns
- Data partition encryption
- Encrypting passwords across a network
- Encrypting data across a network
- SSL host authentication
- Client-side encryption
- Backup file encryption

Data Encryption

- Data Encryption layers

  Application level

  Database level

  Storage

- Two kinds of encryption

  - One way

  - Two way

- Pgcrypto

Pgcrypto

- Extension in PostgreSQL
- Encrypt specific data
- Provides some default functions
- Client Independent

Syntax for extension:

```
create extension pgcrypto;
```

Pgcrypto (continued)

```
CREATE TABLE testusers(username varchar(100) PRIMARY KEY, cryptpwd text, md5pwd text);
INSERT INTO testusers(username, cryptpwd, md5pwd)
    VALUES ('robby', crypt('test', gen_salt('md5')), md5('test')),
        ('artoo', crypt('test',gen_salt('md5')), md5('test'));

SELECT username, cryptpwd, md5pwd
    FROM testusers;

 username |              cryptpwd              |              md5pwd

----------+------------------------------------+----------------------------------
 robby    | $1$IOchfG/z$bZW1pRFA3wuvn6pAuD.Du/ | 098f6bcd4621d373cade4e832627b4f6
 artoo    | $1$84oZTXI/$yZ6wV5jhJo6aQYrTciMQR/ | 098f6bcd4621d373cade4e832627b4f6
```

OS Level Security

- Need to have proper permission to data directory
- Never use 777 (all permission) to any file or directory that is owned by postgres
- Restrict access to configuration files (Postgresql.conf & pg_hba.conf) and log files to unauthorized users.
- Disallow host system login by the iptables.

SQL Injection

- Allows a user to execute arbitrary Structured Query Language (SQL) code to access the database
- Occurs when user input is not filtered for escape characters or executes unexpectedly

  For example, at the login screen for user name and password, a hacker provides a SQL statement or database command (instead of the login name) that goes directly to the database.
- To protect against SQL injection attacks:
  - Check parameters that pass from application
  - When asking for a customer number, check that input is the proper data type, length, etc., before executing the query.
  - Limit the permissions of the account that executes SQL queries.
  - Use stored procedures (or similar techniques) to prevent users from directly interacting with SQL code.

Q & A

Thank You