Learning Objectives

Learners will be able to...

- Describe PostgreSQL host-based authentication
- Recognize and Describe the structure of the authentication file
- List and recall authentication methods
- Identify and troubleshoot common authentication-related error messages in PostgreSQL
- Configure password authentication for a given user to a given database
- Write specific PostgreSQL rules to control database access for users

info

Make Sure You Know

Learners should know how users and roles are managed in PostgreSQL, as this knowledge is foundational to understanding authentication.

Client Authentication

Restricting access to objects within a database is not the only way to set up access restriction. Client authentication is central to PostgreSQL.

info

Authentication

Authentication is the process by which the database server establishes the identity of the client, and by extension determines whether the client application (or the user who runs the client application) is permitted to connect with the database user name that was requested.

Without it, you would either have to sacrifice the ability to connect remotely, or blindly allow everyone who connects to the server to fetch and even modify data. PostgreSQL provides several authentication mechanisms at your disposal.

In PostgreSQL, client access to databases at the host level is specified in the `pg_hba.conf` file. (HBA stands for host-based authentication.)

The entries in this file determine which nodes can connect, the client authentication method, which PostgreSQL usernames they can use, and which cluster databases they can access.

The permissions and restrictions described in this file should not be confused with the database object permissions set for PostgreSQL users. The `pg_hba.conf` file allows you to specify the type of host-level authentication that is performed before connecting to the database, after which user rights take effect.

pg_hba.conf

The `pg_hba.conf` file allows you to specify the type of host-level authentication that is performed before connecting to the database, after which user rights take effect.

info

pg_hba.conf location

The `pg_hba.conf` file is located in the PostgreSQL data directory and is created automatically during installation. Location in current installation is `/etc/postgresql/14/main/pg_hba.conf`.

The pg_hba.conf file consists of lines, and the lines consist of the following fields:

- connection type;
- database name;
- username;
- address(will be omitted for local connection type);
- authentication method;
- optional additional parameters in the form name=value.

Structure example:

```
local    replication     all
peer
host     replication     all             127.0.0.1/32
scram-sha-256
hostssl  replication     all             ::1/128
scram-sha-256
```

Connection types could be following types.
Connection type	Description
———	———-
local	This record matches connection attempts using Unix-domain sockets. Without a record of this type, Unix-domain socket connections are disallowed.
host	This record matches connection attempts made using TCP/IP
hostssl	This record matches connection attempts made using TCP/IP, but only when the connection is made with SSL encryption.

hostnossl	This record type has the opposite behavior of `hostssl`.
hostgssenc	This record matches connection attempts made using TCP/IP, but only when the connection is made with GSSAPI encryption.
hostnogssenc	This record type has the opposite behavior of `hostgssenc`.

Database name rules:
- **all** - connection to any database;
- **sameuser** - database with the same name as the role;
- **samerole** - a database that has the same name as the role or group it belongs to;
- **replication** - special permission for the replication protocol;
- **exact database name** - the name of a specific database, or you can list a list of databases separated by commas.

Username rules:

- **all** - any available role
- **exact role name** - role with the specified name, while you can list the roles separated by commas

Address rules:

- **all** - any IP address;
- **ip address** - specific ip address, subnet, or range (172.20.143.0/24 or 172.20.143.0 255.255.255.0)
- **samehost** - means the address of the server from which the connection is made, analogous to 127.0.0.1;
- **samenet** - means any ip address from the server subnet;
- **domain name** (or part of the domain name, starting with a dot) - while PostgreSQL checks the ip-address of the connecting client for belonging to this domain.

Authentication Methods

PostgreSQL provides a wide range of authentication methods, the most commonly used are the following:

- **Trust authentication**, which simply trusts that users are who they say they are.
- **Password authentication**, which requires that users send a password.
- **Peer authentication**, which relies on operating system facilities to identify the process at the other end of a local connection. This is not supported for remote connections.

A complete list of methods can be found in the documentation.

Trust authentication

When `trust` authentication is specified, PostgreSQL assumes that anyone who can connect to the server is authorized to access the database with whatever database user name they specify (even superuser names). Of course, restrictions made in the database and user columns still apply. This method should only be used when there is adequate operating-system-level protection on connections to the server.

```
# TYPE    DATABASE    USER    ADDRESS    METHOD
host      all         all     all        trust
```

Opposite method is `reject` - deny all connection.

```
# TYPE    DATABASE    USER    ADDRESS    METHOD
host      all         all     all        reject
```

Password authentication

There are several password-based authentication methods. These methods operate similarly but differ in how the users' passwords are stored on the server and how the password provided by a client is sent across the connection.

Type	Description
scram-sha-256	The method will check the password provided by the user using SCRAM-SHA-256 authentication. This is the most secure of the currently

	provided methods, but it is not supported by older client libraries.
md5	The method uses a custom less secure challenge-response mechanism.
password	The method sends the password in clear-text and is therefore vulnerable to password “sniffing” attacks.

Entry example:

```
# TYPE    DATABASE    USER    ADDRESS    METHOD
host      all         all     all        scram-sha-256
```

Peer Authentication

The peer authentication method works by obtaining the client’s operating system user name from the kernel and using it as the allowed database user name (with optional user name mapping). This method is only supported on local connections.

Entry example:

```
# TYPE    DATABASE    USER    ADDRESS    METHOD
local     all         all                peer
```

Authentication Problems

Authentication failures and related problems generally manifest themselves through error messages like the following:

```
FATAL:  no pg_hba.conf entry for host "127.0.0.1", user
"my_user", database "codio", no encryption
```

This is what you are most likely to get if you succeed in contacting the server, but it does not want to talk to you. As the message suggests, the server refused the connection request because it found no matching entry in its `pg_hba.conf` configuration file.

```
FATAL:  password authentication failed for user "my_user"
```

Messages like this indicate that you contacted the server, and it is willing to talk to you, but not until you pass the authorization method specified in the pg_hba.conf file. Check the password you are providing, or check your Kerberos or ident software if the complaint mentions one of those authentication types.

```
FATAL:  user "my_user" does not exist
```

The indicated database user name was not found.

```
FATAL:  database "testdb" does not exist
```

The database you are trying to connect to does not exist. Note that if you do not specify a database name, it defaults to the database user name, which might or might not be the right thing.