Database systems are rich with attack vectors to exploit. This presentation explores the many potential PostgreSQL external vulnerabilities and shows how they can be secured. *Includes concepts from Magnus Hagander*
Attack Vectors

https://www.flickr.com/photos/twalmsley/
External Attack Vectors

- 'Trust' security
- Passwords / authentication theft
- Network snooping
- Network pass-through spoofing
- Server / backup theft
- Administrator access
Internal Attack Vectors
(Not Covered)

- Database object permissions
- SQL injection attacks
- Application vulnerability
- Operating system compromise
Authentication Security

https://www.flickr.com/photos/brookward/
Avoid “Trust” Security in pg_hba.conf

# TYPE DATABASE USER CIDR-ADDRESS METHOD
# "local" is for Unix domain socket connections only
local all all trust
# IPv4 local connections:
host all all 127.0.0.1/32 trust
# IPv6 local connections:
host all all ::1/128 trust

Solution: Use the initdb -A flag, i.e., you don’t want to see this:

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the -A option the next time you run initdb.
Password Snooping

Using 'username' in the MD5 string prevents the same password used by different users from appearing the same. It also adds some randomness to the md5 checksums.
MD5 Authentication
Prevents Password Snooping

- Database
  - Client
    - connection request
    - need password, sent random salt
    - $md5(md5(password+username) + salt)$

- PostgreSQL
  - Database
    - Server
      - md5(password+username)
      - md5(password+username)
      - md5(password+username)
      - md5(password+username)
**MD5 Authentication**

Prevents Password Replay

---

*salt* is a random four-byte integer so millions of connection attempts might allow the reuse of an old authentication reply.
SCRAM Authentication

scatter-sha-256, available in Postgres 10, eliminates less-secure MD5, and avoids the risk of duplicate salt values being replayed. SCRAM with channel binding, available in Postgres 13, allows authentication, similar to certificate authentication.
Password Attacks

- Weak passwords
- Reuse of old passwords
- Brute-force password attacks

None of these vulnerabilities is prevented by Postgres directly, but external authentication methods, like LDAP, PAM, and SSPI, can prevent them. Some authentication methods are difficult to use with connection pooling.
Queries and Data Still Vulnerable to Network Snooping

Password changes are also vulnerable to snooping.
SSL Prevents Snooping
By Encrypting Queries and Data

Queries and data encrypted by SSL

Database Client

AES256(SELECT * FROM customer);

AES256(Barr Bearings | $10230 | James Akel)

Queries and data encrypted by SSL

PostgreSQL
Database Server
Preventing Spoofing

https://www.flickr.com/photos/tomhickmore/
Localhost Spoofing
While the Database Server Is Down

The server controls the choice of 'password' instead of 'md5'.

Uses a fake socket or binds to port 5432 while the real server is down. (/tmp is world-writable and 5432 is not a root-only port. libpq's "requirepeer" helps here.)
Network Spoofing

Without SSL 'root' certificates there is no way to know if the server you are connecting to is a legitimate server.
Network Spoofing Pass-Through

Records passwords for later use with the real server. It can also capture queries, data, and inject its own queries.

Without SSL 'root' certificates there is no way to know if the server you are connecting to is a legitimate server.
SSL ’Prefer’ Is Not Secure

Without SSL ‘root’ certificates there is no way to know if the server you are connecting to is a legitimate server.
SSL ’Require’ Is Not Secure From Spoofing

Without SSL ‘root’ certificates there is no way to know if the server you are connecting to is a legitimate server.

Records passwords for later use with the real server. It can also capture queries, data, and inject its own queries.

Database Client

Fake PostgreSQL

Database Server

PostgreSQL

Database Server

SSL or Non-SSL

OK SSL

SSL

Query

Result

Query

Result
SSL ’Verify-CA’ Is Secure From Spoofing

Database Client

x

root.crt

SSL verify-ca

Invalid certificate
(no CA signature)

Fake PostgreSQL
Database
Server

PostgreSQL
Database
Server

server.crt
SSL Certificates for Authentication

Database Client

request for certificate

SSL certificate w/ cn

PostgreSQL

Database

Server

root.crt

server.crt
Data Encryption
To Avoid Data Theft

https://www.flickr.com/photos/debarshiray/
Disk Volume Encryption

This helps prevent stolen storage devices from being read, and helps with secure media destruction.

https://www.flickr.com/photos/icebrkr/
### Column Encryption

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>credit_card_number</th>
</tr>
</thead>
<tbody>
<tr>
<td>428914</td>
<td>Piller Plaster Co.</td>
<td>\xc30d04070302254dc045353f28; 456cd241013e2d421e198f3320e8; 41a7e4f751ebd9e2938cb6932390; 5c339c02b5a8580663d6249eb24f; 192e226c1647dc02536eb6a79a65; 3f3ed455ff5726ca2b67430d5</td>
</tr>
</tbody>
</table>

Encryption methods are decryptable (e.g., AES), while hashes are one-way (e.g., MD5). A one-way hash is best for data like passwords that only need to be checked for a match, rather than decrypted.
Where to Store the Key?
On the Server

SELECT * FROM customer;

Barr Bearings | $10230 | James Akel
Decrypted data
Store the Key on an Intermediate Server

- Database Client
- Cryptographic Server
- PostgreSQL Database Server

SELECT

Barr Bearings
Decrypted

Encrypted

V#ja20a

26 / 30
Store the Key on the Client and Encrypt/Decrypt on the Server

SELECT decrypt(col, key) FROM customer;

Barr Bearings | $10230 | James Akel

Decrypted data
Encrypt/Decrypt on the Client

This prevents server administrators from viewing sensitive data.
Store the Key on a Client Hardware Token

This prevents problems caused by client hardware theft.
Conclusion

https://momjian.us/presentations

https://www.flickr.com/photos/stevensnodgrass/