Securing PostgreSQL From External Attack

Bruce Momjian,
EnterpriseDB

September, 2009

Abstract
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.

Creative Commons Attribution License
http://momjian.us/presentations
Attack Vectors
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
- Application vulnerability
- Operating system compromise
Authentication Security

http://www.my-time-machines.net/mosler_34.htm
Avoid ‘Trust’ Security

# 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.

Securing PostgreSQL From External Attack
MD5 Authentication
Prevents Password Snooping

Database
Client

connection request
need password, sent random salt
md5(md5(username+password) + salt)

PostgreSQL
Database
Server
md5(username+password)
md5(username+password)
md5(username+password)
md5(username+password)
MD5 Authentication
Prevents Password Replay

\[
\text{connection request} \\
\text{need password, sent random salt0} \\
\text{md5(md5(username+password) + salt0)} \\
\text{ok} \\
\text{need password, sent random salt1} \\
\text{md5(md5(username+password) + salt0)} \text{ (replay)} \\
\text{x}
\]

\text{salt} \text{ is a random four-byte integer so millions of connection attempts might allow the reuse of an old authentication reply.}
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.
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 customers);

AES256(Barr Bearings | $10230 | James Akel)

PostgreSQL Database Server
Preventing Spoofing

http://redwing.hutman.net/~mreed/warriorshtm/impostor.htm
Localhost Spoofing
While the Database Server Is Down

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

Securing PostgreSQL From External Attack
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

Connection Request
Need Plain Password
Password Sent
Query
Result

Fake PostgreSQL

Database

Server

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

OK

PostgreSQL

Database

Server

Query
Result

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

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 ’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.
SSL ‘Verify-CA’ Is Secure From Spoofing

Database Client

SSL verify-ca
Invalid certificate (no CA signature)

Fake PostgreSQL
Database
Server

PostgreSQL
Database
Server

Securing PostgreSQL From External Attack
SSL ’Verify-full’ Is Secure Even From Some Certificate Thefts

Certificate stolen from a CA–trusted computer, but not the database server.
Data Encryption
To Avoid Data Theft

http://jproc.ca/crypto/enigma.html
Disk Volume Encryption
## 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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; 456cd241013e2d421e198f3320e8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; 41a7e4f751ebd9e2938cb6932390</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; 5c339c02b5a8580663d6249eb24f</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; 192e226c1647dc02536eb6a79a65</td>
</tr>
<tr>
<td></td>
<td></td>
<td>; 3f3ed455ffc5726ca2b67430d5</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 customers;
Barr Bearings | $10230 | James Akel
Decrypted data

Securing PostgreSQL From External Attack 24
Store the Decryption Key On an Intermediate Server

SELECT

Database Client

Cryptographic Server

key

SELECT

PostgreSQL Database Server

Barr Bearings Encrypted

Decrypted

Securing PostgreSQL From External Attack
Pass the Decryption Key Via SQL From the Client

```
SELECT decrypt(col, key) FROM customers;
```

Barr Bearings | $10230 | James Akel

Decrypted data
Decrypt On the Client

SELECT * FROM customers;

V#aei32ok3

Encrypted data

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

This prevents problems caused by client hardware theft.
Conclusion