Best Practices in Security with PostgreSQL

Dave Page

May 2021

Welcome – Housekeeping Items

- Slides and recording will be available in next 48 hours
- Submit questions via the question panel – will be answering at end

Dave Page

- EDB (CTO Office)
  - VP & Chief Architect, Database Infrastructure

- PostgreSQL
  - Core Team
  - pgAdmin Lead Developer
  - PostgreSQL Europe (Secretary)
  - PostgreSQL Community Association of Canada (Chairperson)

Agenda

- Introduction to EDB
- Aspects of Data Security
- General recommendations
- Overall Framework and today's focus
- Key Concepts: Authentication, Authorization, Auditing
- Data encryption
- Summary
- Q&A

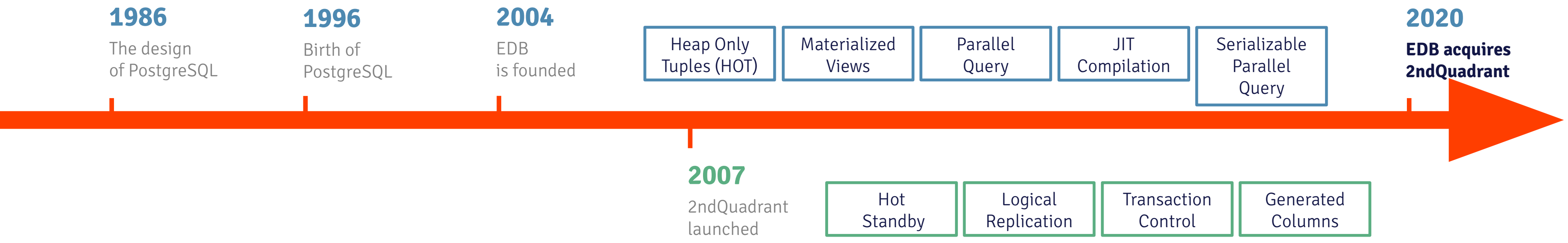
Expertise

We’re database fanatics who care deeply about PostgreSQL

Enterprise PostgreSQL innovations

PostgreSQL community leadership

Recognized by Gartner and Forrester



The most PostgreSQL experts

The EDB team includes:

- 300+ PostgreSQL technologists
- 26 PostgreSQL community contributors and committers, including founders and leaders including:

Michael Stonebraker
"Father of Postgres"
and EDB Advisor

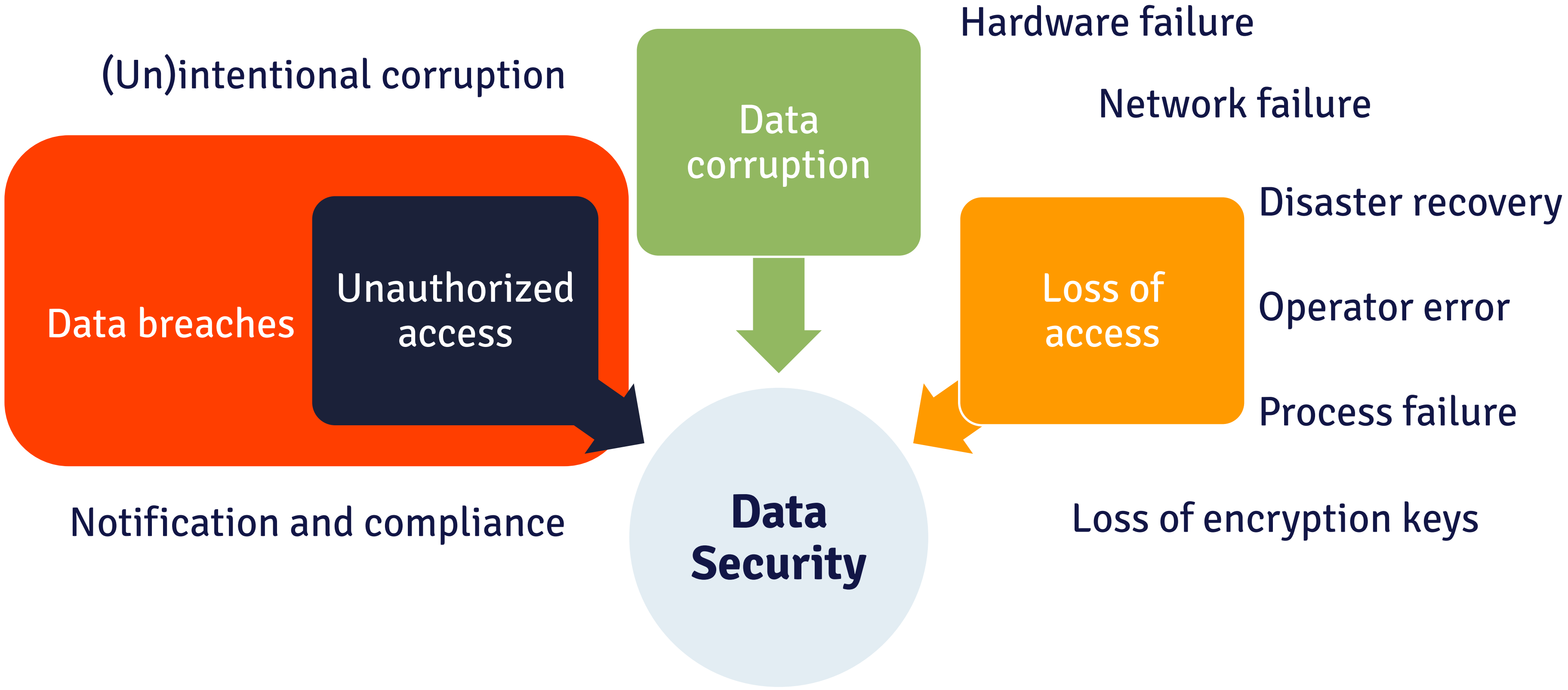
Bruce Momjian
Co-founder, PostgreSQL
Global Development Group
and PostgreSQL Core Team

Peter Eisentraut
PostgreSQL
Core Team member

Robert Haas
PostgreSQL Major
Contributor and Committer

Simon Riggs
PostgreSQL Major Contributor,
Founder
of 2ndQuadrant

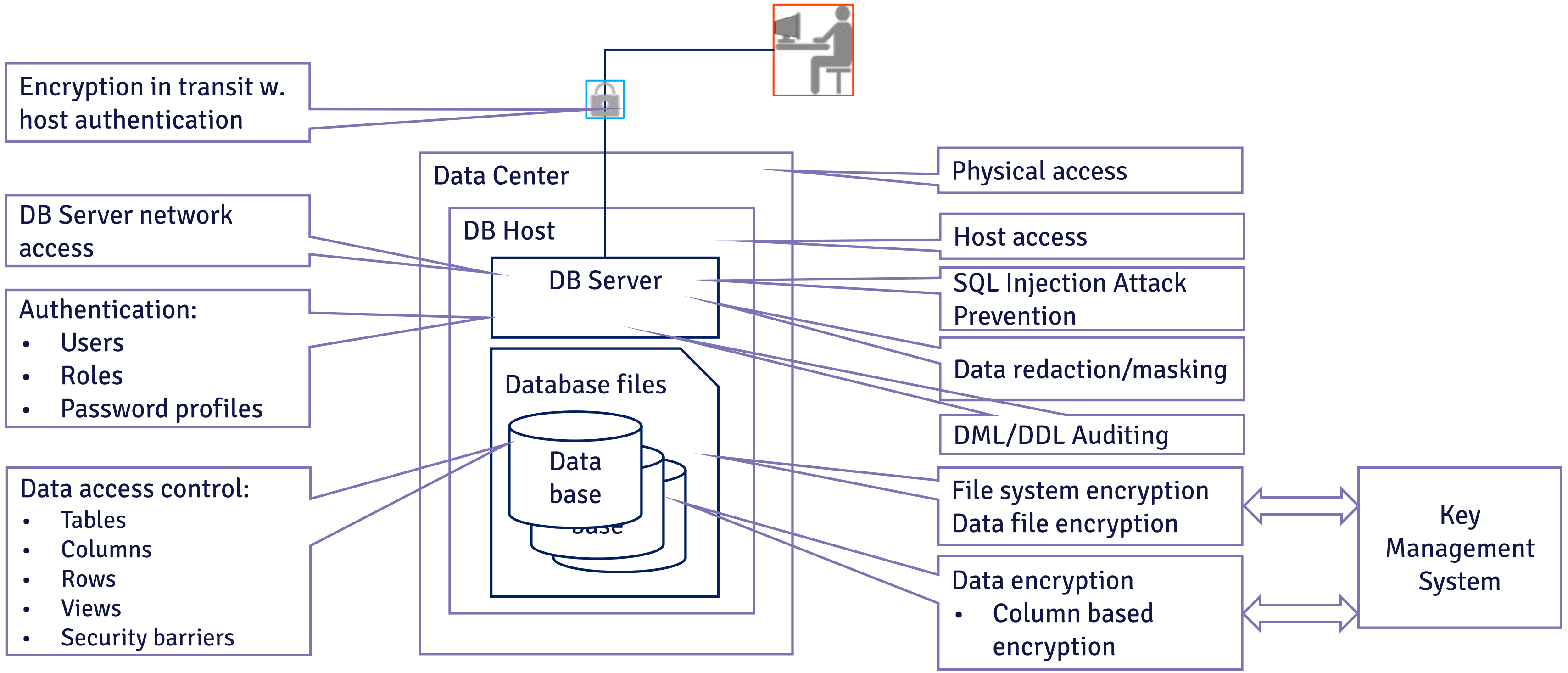
Aspects of Data Security



General Recommendations

- Keep your operating system and your database patched
- Don’t put a postmaster port on the internet
- Isolate the database port from other network traffic
- Grant users the minimum access they require to do their work, nothing more
- Restrict access to configuration files (postgresql.conf and pg_hba.conf)
- Disallow host system login by the database superuser roles
- Provide each user with their own login
- Don’t rely solely on your front-end application to prevent unauthorized access
- Keep backups, and have a tested recovery plan

Multiple layers of security



Today’s Focus

- Access to the database application
- Access to the data contained within the database
- Secure the data stored in the database

AAA Model

Popular model for security architectures

- **Authentication:** verify that the user is who they claim to be
- **Authorization:** verify that the user is allowed access
- **Auditing (or Accounting):** record all database activity, including the user name and the time in the log files

Authentication

Defined in hba.conf ⇐ make sure you understand how this works and protect that file!

- Kerberos/GSSAPI Single Sign-On (SSO) authentication
  - data sent over the database connection is unencrypted unless SSL or GSS encryption is in use
- SSPI – Windows Single Sign-On (SSO) authentication
- LDAP and ~~RADIUS~~
  - LDAP (specifically, LDAP+STARTTLS) should only be used if Kerberos is out of the question.
  - LDAP passwords are forwarded to the LDAP server, and it can easily be set up in an insecure way
  - ~~RADIUS should not be used because it has weak encryption, using md5 hashing for credentials~~
- Cert – TLS certificate authentication; often used in machine-to-machine communication
- ~~md5~~ and scram – stores username and password information in the database
  - Scram is highly preferred over md5 as the passwords are securely hashed
  - Use with EDB Postgres password profiles

Password Profiles

EDB Postgres Advanced Server 9.5 and above

Oracle compatible password profiles can be used to:

- specify the number of allowable failed login attempts
- lock an account due to excessive failed login attempts
- mark a password for expiration
- define a grace period after a password expiration
- define rules for password complexity
- define rules that limit password reuse

Password Profiles - Setup (1 of 4)

```
-- Create profile and a user
CREATE PROFILE myprofile;
CREATE USER myuser IDENTIFIED BY mypassword;
-- Assign profile to a user
ALTER USER myuser PROFILE myprofile;

-- Check the user-profile mapping
SELECT rolname, rolprofile FROM pg_roles WHERE rolname = 'myuser';
 rolname | rolprofile
---------+------------
 myuser  | myprofile
```

Password Profiles - Definition of Rules (2 of 4)

```
ALTER PROFILE myprofile LIMIT
  FAILED_LOGIN_ATTEMPTS 3
  PASSWORD_LOCK_TIME 2;
```

```
SELECT rolname, rolprofile, edb_get_role_status(oid), rolfailedlogins, rollockdate FROM pg_roles
WHERE rolname = 'myuser';
 rolname | rolprofile | edb_get_role_status | rolfailedlogins | rollockdate
---------+------------+---------------------+-----------------+-------------
 myuser  | myprofile  | OPEN                |               0 |
```

Password Profiles - 1st failed login (3 of 4)

```
\c - myuser
Password for user myuser:
FATAL:  password authentication failed for user "myuser"

SELECT rolname, rolprofile, edb_get_role_status(oid), rolfailedlogins, rollockdate FROM pg_roles
WHERE rolname = 'myuser';
 rolname | rolprofile | edb_get_role_status | rolfailedlogins | rollockdate
---------+------------+---------------------+-----------------+-------------
 myuser  | myprofile  | OPEN                |               1 |
```

Password Profiles - Account Locked (4 of 4)

```
\c - myuser
Password for user myuser:
FATAL:  role "myuser" is locked
Previous connection kept

SELECT rolname, rolprofile, edb_get_role_status(oid), rolfailedlogins, rollockdate FROM pg_roles
WHERE rolname = 'myuser';
 rolname | rolprofile | edb_get_role_status | rolfailedlogins |           rollockdate
---------+------------+---------------------+-----------------+----------------------------------
 myuser  | myprofile  | LOCKED(TIMED)       |               0 | 13-NOV-18 12:25:50.811022 +05
```

Super user interaction

```
ALTER USER myuser ACCOUNT UNLOCK;
```

Authorization

We know who you are - what are you allowed to do?

- Standard method: Manage access privileges to tables, views and other objects
- Best Practice:
  - Revoke CREATE privileges from all users and grant them back to trusted users only
  - Don't allow the use of functions or triggers written in untrusted procedural languages
  - SECURITY DEFINER functions ⇐ understand what that means
  - Database objects should be owned by a secure role
- Beware: when log_statement is set to 'ddl' or higher, ALTER ROLE command can result in password exposure in the logs, except in EDB Postgres Advanced Server 11+
  - Use **edb_filter_log.redact_password_command** to redact stored passwords from the log file

Row Level Security (a.k.a. Virtual Private Database)

Restrict, on a per-user basis, which rows can be returned by normal queries or inserted, updated, or deleted by data modification commands

```
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);
```

DBMS_RLS provides key functions for Oracle's Virtual Private Database in EDB Postgres Advanced Server

Data Redaction

```
Username [enterprisedb]: privilegeduser
mycompany=> select * from employees;
id | name          | ssn         |   phone     |   birthday
----+--------------+-------------+------------+--------------------
 1 | Sally Sample | 020-78-9345 | 5081234567 | 02-FEB-61 00:00:00
 1 | Jane Doe     | 123-33-9345 | 6171234567 | 14-FEB-63 00:00:00
 1 | Bill Foo     | 123-89-9345 | 9781234567 | 14-FEB-63 00:00:00
(3 rows)
```

```
Username [enterprisedb]: redacteduser
mycompany=> select * from employees;
id | name          | ssn         |   phone     |   birthday
----+--------------+-------------+------------+--------------------
 1 | Sally Sample | xxx-xx-9345 | 5081234567 | 02-FEB-02 00:00:00
 1 | Jane Doe     | xxx-xx-9345 | 6171234567 | 14-FEB-02 00:00:00
 1 | Bill Foo     | xxx-xx-9345 | 9781234567 | 14-FEB-02 00:00:00
(3 rows)
```

Auditing

EDB Postgres Advanced Server offers enhanced auditing

- Track and analyze database activities
- Record connections by database Users
  - Successful and failed
- Record SQL activity by database Users
  - Errors, rollbacks, all DDL, all DML, all SQL statements
- Session Tag Auditing
  - Associate middle-tier application data with specific activities in the database log (e.g. track application Users or IP addresses not just database users)

Audit Configuration Params

- postgresql.conf parameter: edb_audit (Values = XML or CSV)
- edb_audit_directory & edb_audit_filename
- edb_audit_rotation_day, edb_audit_rotation_size, edb_audit_rotation_seconds
- edb_audit_connect and edb_audit_disconnect
- edb_audit_statement
  - Specifies which SQL statements to capture

```
edb_audit_connect = 'all'
edb_audit_statement = create view,create materialized view,create
sequence,grant'
```

- edb_filter_log.redact_password_commands ⇐ **Redacts passwords from audit file!!!**

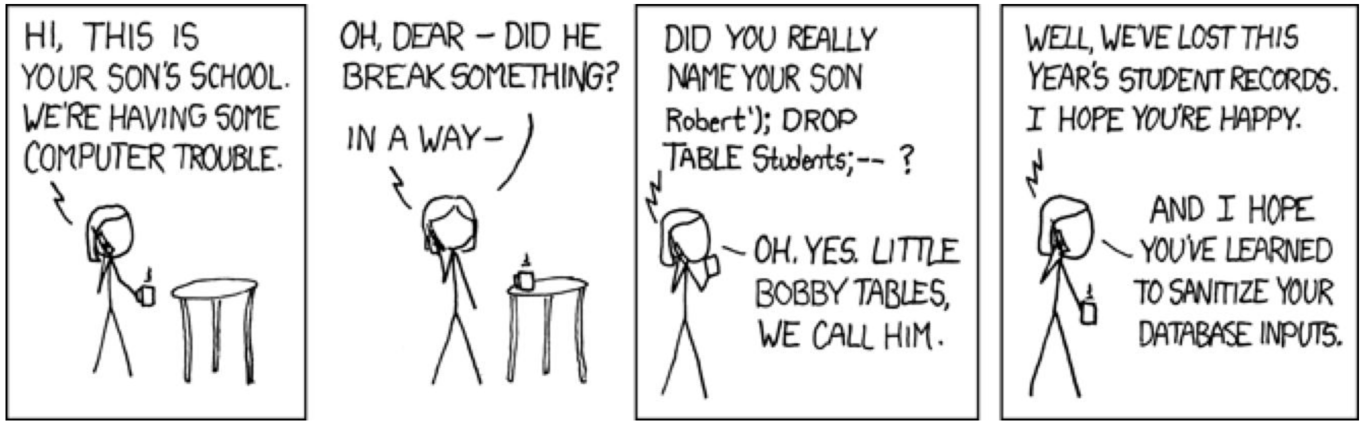
Encryption

Encrypt at rest and in transit -- key: Understand the threat vector!

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

SQL Injection Prevention

- SQL Injection attacks are possible where applications are designed in a way that allows the attacker to modify SQL that is executed on the database server
- By far the most common way to create a vulnerability of this type is by creating SQL queries by concatenating strings that include user-supplied data


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

From: https://www.explainxkcd.com/wiki/index.php/327:_Exploits_of_a_Mom

SQL Injection Prevention

Example

- Consider a website which will login a user using a query constructed as follows:

```
login_ok = conn.execute("SELECT count(*) FROM users WHERE name = '" + username + "'
AND password = '" + password + "';");
```

- If the user enters their username as `dave` and their password as `secret' OR '1' = '1`, the generated SQL will become:

```
SELECT count(*) FROM users WHERE name = 'dave' AND password = ' secret' OR '1' =
'1';
```

- If the code is testing that login_ok has a non-zero value to authenticate the user, then the user will be logged in regardless of whether the username/password is correct.

SQL Injection Prevention

Protecting against it in the application - sanitize the user input!

- Don't use string concatenation to include user supplied input in queries!
- Use parameterised queries instead, and let the language, driver, or database handle it.
- Here's a Python example (using the psycopg2 driver):

```
cursor.execute("""SELECT count(*) FROM users WHERE username = %s AND
password = %s;""", (username, password))
```

SQL Protect

EDB Postgres Advanced Server: Additional SQL Injection Prevention at the Database Level

- Utility Commands
  - Any DDL commands: DROP TABLE
- SQL Tautologies
  - SQL WHERE predicates such as… and 1=1
- Empty DML
- DML commands with no WHERE filter, such as: DELETE FROM EMPLOYEE;
- Unauthorized Relations
  - Results from Learn mode associating roles with tables

Conclusion

Security comes in layers!

AAA (Authorization, Authentication, Auditing) reference model

Encryption at rest and on the wire has to be part of the plan

Least privilege approach is key

Read, read, and read some more!

- EDB Security Technical Implementation Guidelines (STIG) for PostgreSQL on Windows and Linux
- Blog: How to Secure PostgreSQL: Security Hardening Best Practices & Tips
- Blog: Managing Roles with Password Profiles: Part 1
- Blog: Managing Roles with Password Profiles: Part 2
- Blog: Managing Roles with Password Profiles: Part 3

Thank You