Raj Verma
Chief Operating Officer
CYBERTEC

www.cybertec-postgresql.com
@cybertec-postgresql
www.youtube.com/@cybertecpostgresql

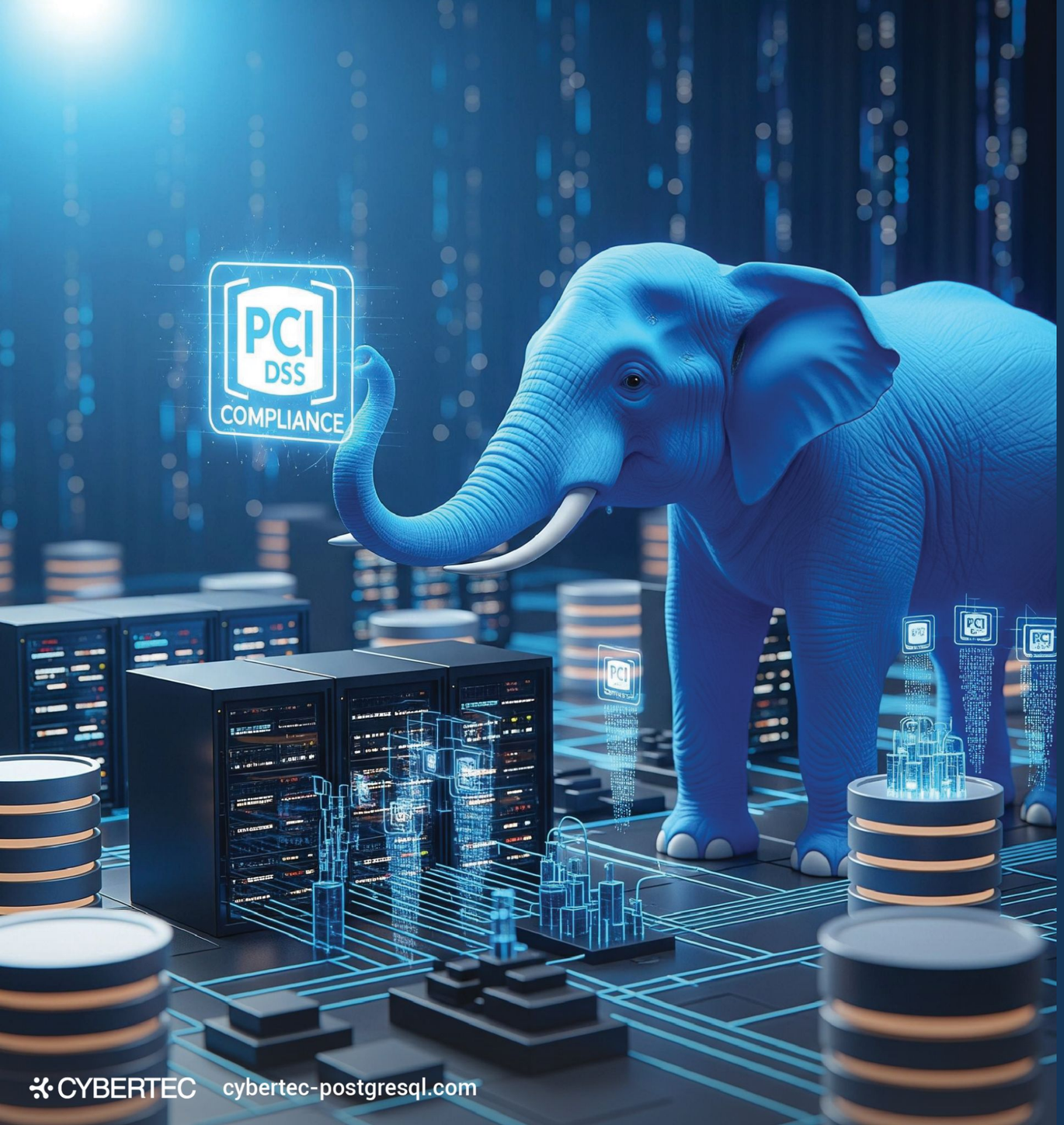
CYBERTEC

Database excellence

- 25 years of experience
- Operating worldwide
- Independent, professional, and honest

From Chaos to Compliance

How PostgreSQL Makes Regulations Work for You



Why does it matter?

“Compliance is everywhere”

Growing Regulatory Pressure:

Industries like finance (PCI DSS), healthcare (HIPAA), and government (GDPR, FedRAMP) require strict data security controls.

Cost of Non-Compliance:

Fines can reach millions, not to mention loss of customer trust.

Traditional Solutions Are Complex:

Many databases require third-party tools, custom scripts, and manual processes to meet compliance—adding cost and risk.

Different Compliance types

- PCI-DSS
- ISO27001
- GDPR
- TISAX
- HIPAA
- SOX ……. and the list goes on.

Real-world patterns!

PCI DSS: What it is …

Credit card standard

Important standard

Widely accepted worldwide

Various levels

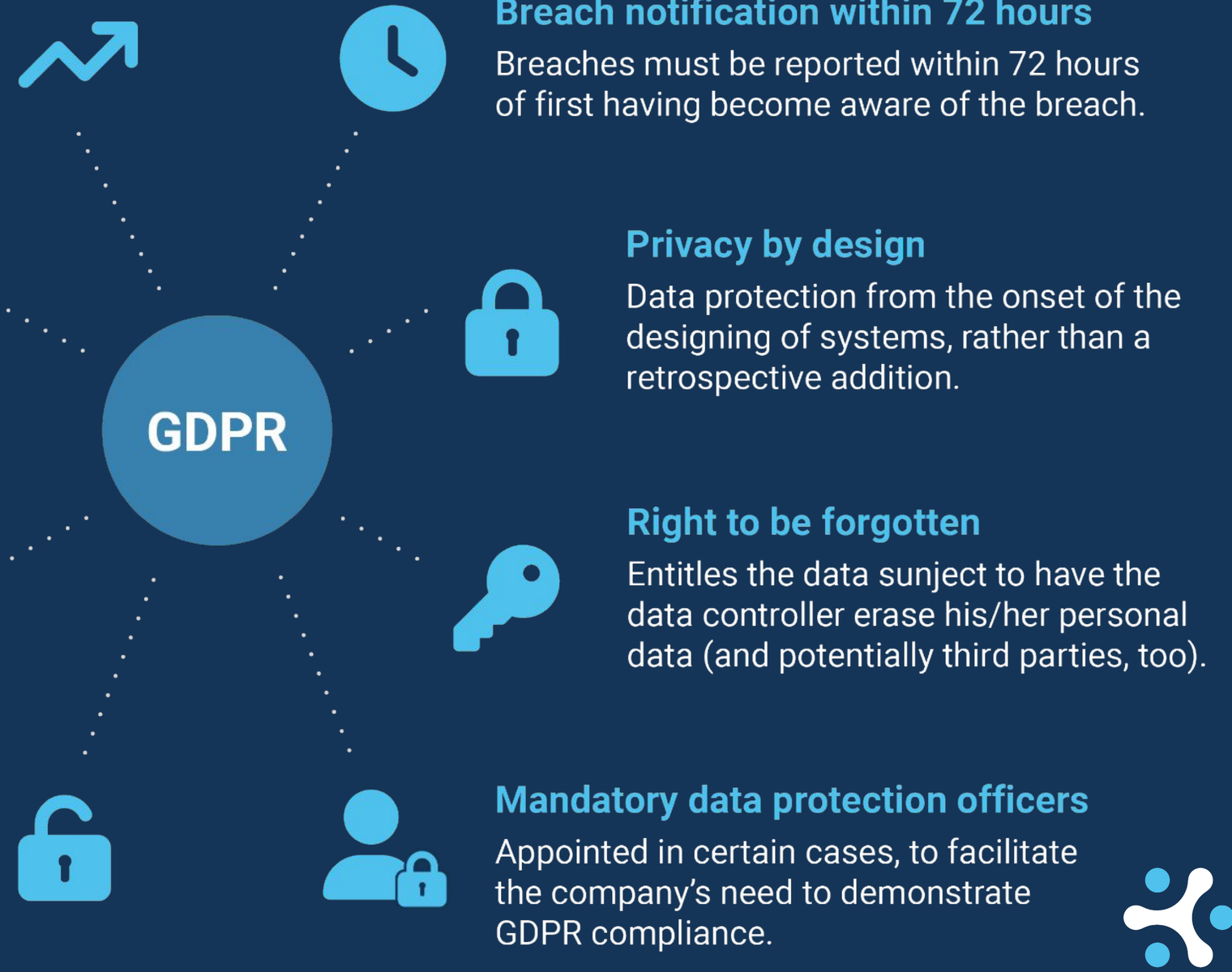
Depending on volume

PCI DSS Compliance Levels	
Level 1	6M + Transactions / Year
Level 2	1- 6M Transactions / Year
Level 3	20K - 1M Transactions / Year
Level 4	< 20K Transactions / Year

Audit requirements

- Different from level to level
- The more transactions, the more requirement

Bigger Responsibility, Bigger Repercussions



Fines of up to 4% of turnover

Organizations in breach of GDPR can be fined up to 4% of annual global turnover or £20 Million.

Breach notification within 72 hours

Breaches must be reported within 72 hours of first having become aware of the breach.

Increased territorial scope

Applies to any company processing personal data of EU citizens, regardless of location

Privacy by design

Data protection from the onset of the designing of systems, rather than a retrospective addition.

Consent matters

Explicit consent must be provided in an intelligible and easily accessible form.

Right to be forgotten

Entitles the data sunject to have the data controller erase his/her personal data (and potentially third parties, too).

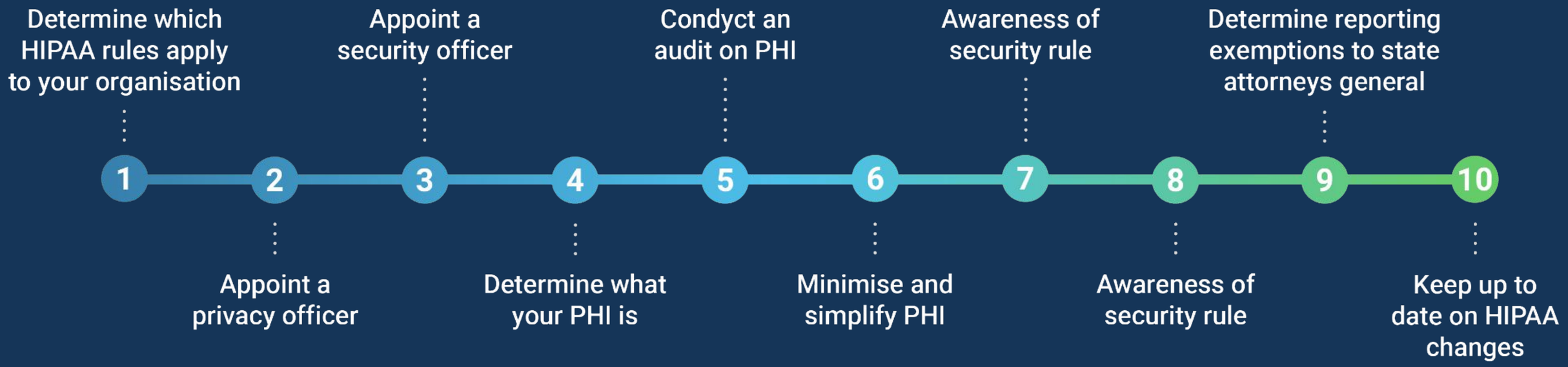
Right to access and portability

Users can inquire whether and how their personal data is being processed.

Mandatory data protection officers

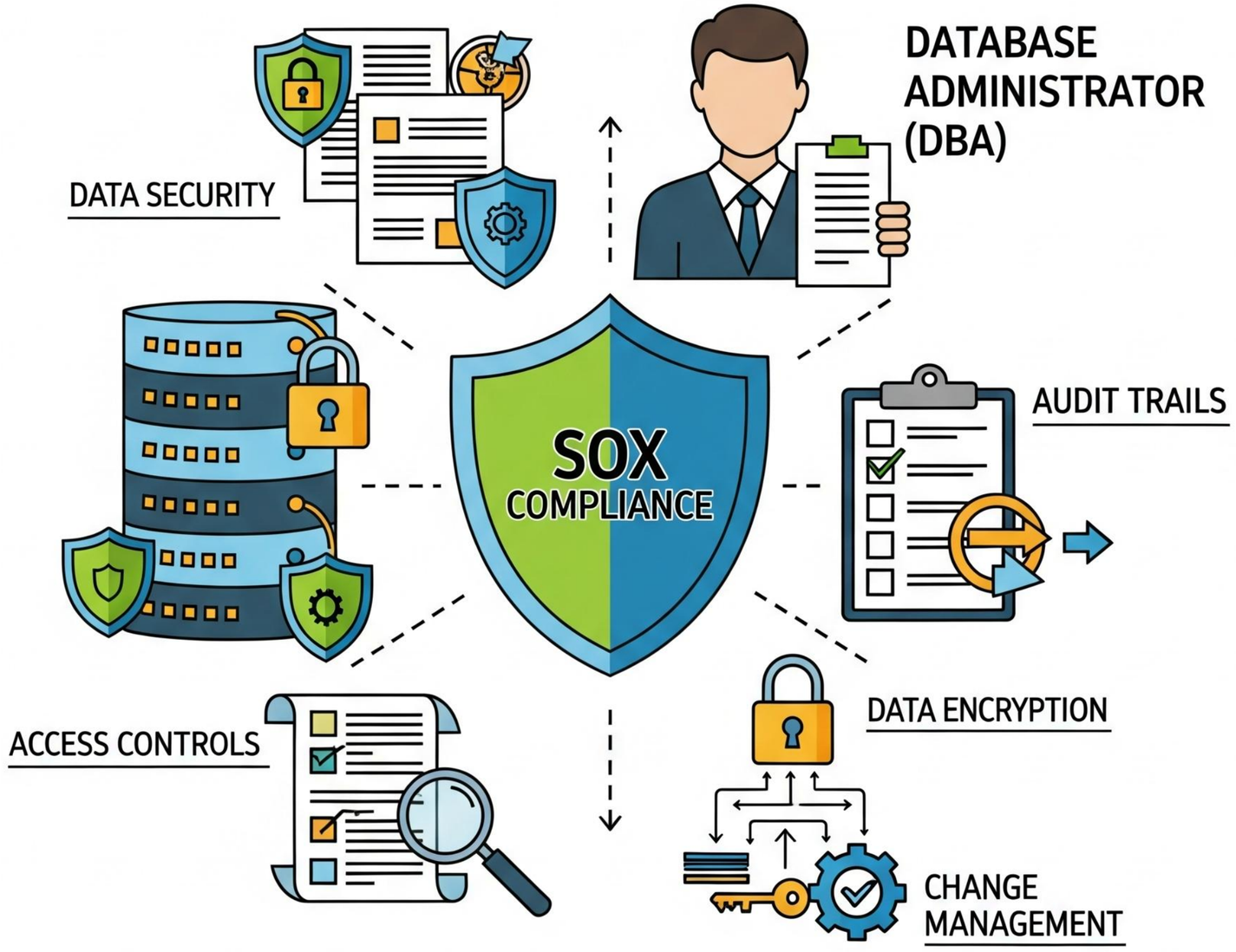
Appointed in certain cases, to facilitate the company's need to demonstrate GDPR compliance.

10 steps to HIPAA compliance



CYBERTEC

CYBERTEC 10 Steps to HIPAA Compliance | Source: THE HIPAA JOURNAL 2025

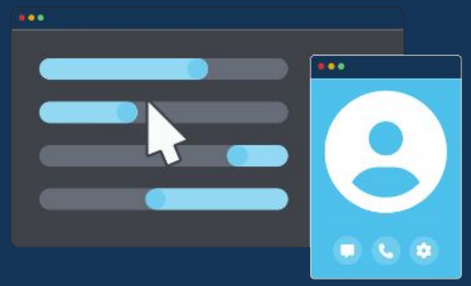

DATABASE ADMINISTRATOR (DBA)
DATA SECURITY
SOX COMPLIANCE
AUDIT TRAILS
ACCESS CONTROLS
DATA ENCRYPTION
CHANGE MANAGEMENT

Access Controls

Audit Trails

Access Controls

SOX
Compliance



Database
Administrator
(DBA)

Change Management

Data Encryption


TO:DO.
FOCUS!
get sh*t
in
order

House of Commons

1. Access Control / Authorization
2. Authentication / Identity Assurance
3. Encryption / Confidentiality
   (Data At Rest & In Transit)
4. Auditing / Logging / Monitoring
5. Integrity / Change Control / Detection
   of Unauthorized Changes
6. Data Minimization, Retention, & Disposal
7. Network Segmentation & Scope Reduction
8. Risk Assessment & Management
9. Incident Response

How PostgreSQL features & extensions make compliance easier than you think!

IAM

Identity & Access Management


RESTRICTED
ACCESS
NEED-TO-KNOW BASIS ONLY

1. Say “NO” to trust by default.
2. “Comments matter”
3. Only allows SSL-encrypted connections.
4. Restrict attack surface.
5. Everything else is rejected.

pg_hba.conf & listen_addresses

```
# TYPE  DATABASE  USER        ADDRESS                 METHOD   OPTIONS
hostnossl all       all         all                   reject
```

```
# Local connections
local   all         postgres                          peer
local   all         all                               peer

# Allow only trusted IPs or subnets with SSL required
hostssl all         all         192.168.10.0/24       md5
hostssl all         all         10.20.0.15/32         scram-sha-256
hostssl all         all         203.0.113.42/32       md5

# Block everything else
```

```
# postgresql.conf
listen_addresses = '127.0.0.1,192.168.10.5'
```

```
postgres=# show unix_socket_permissions;
 unix_socket_permissions
-------------------------
 0777
(1 row)

postgres=#
postgres=# show unix_socket_directories;
 unix_socket_directories
-------------------------
 /tmp
(1 row)
```

```
➜  /tmp la
total 16
srwxrwxrwx@ 1 rajverma  wheel     0B Oct 15 12:15 .s.PGSQL.5432
-rw-------@ 1 rajverma  wheel    75B Oct 15 12:15
.s.PGSQL.5432.lock
```

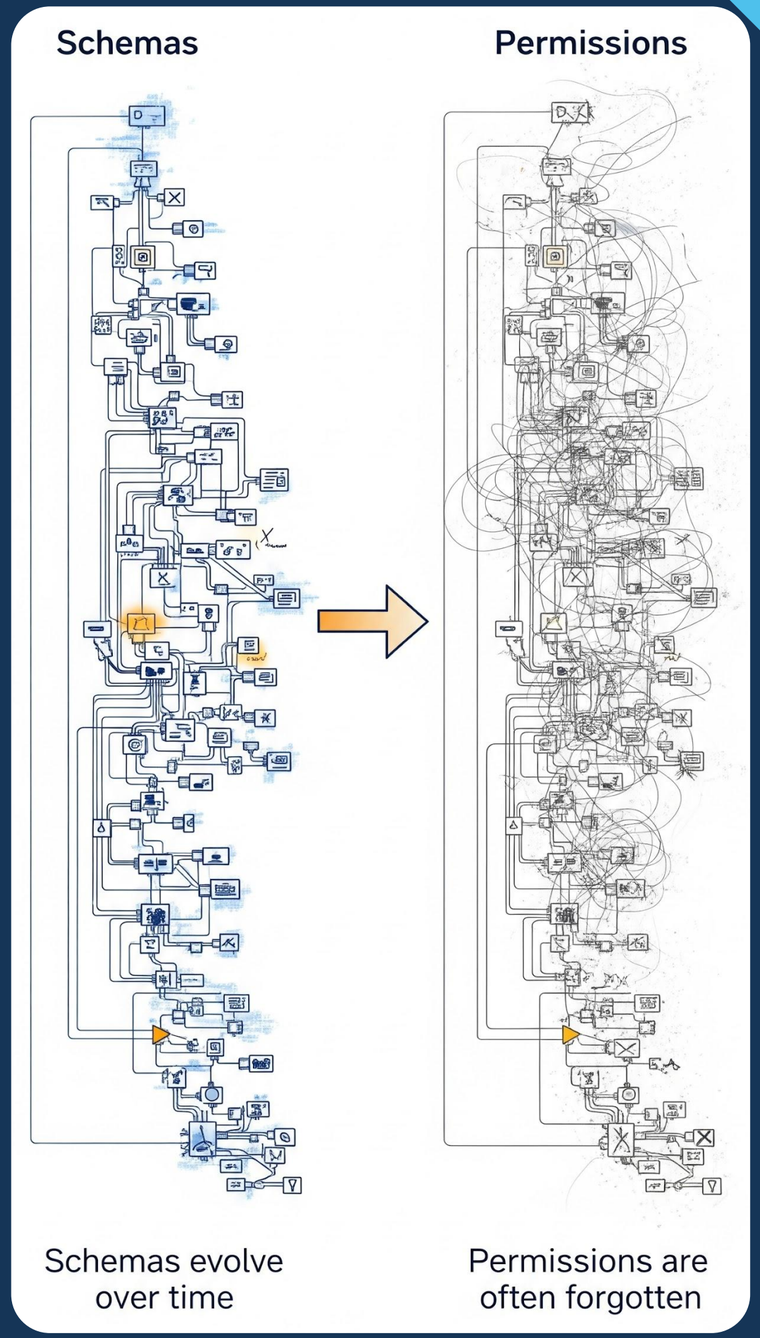
I am G“root”

Alter Default Privileges

Setting Standards

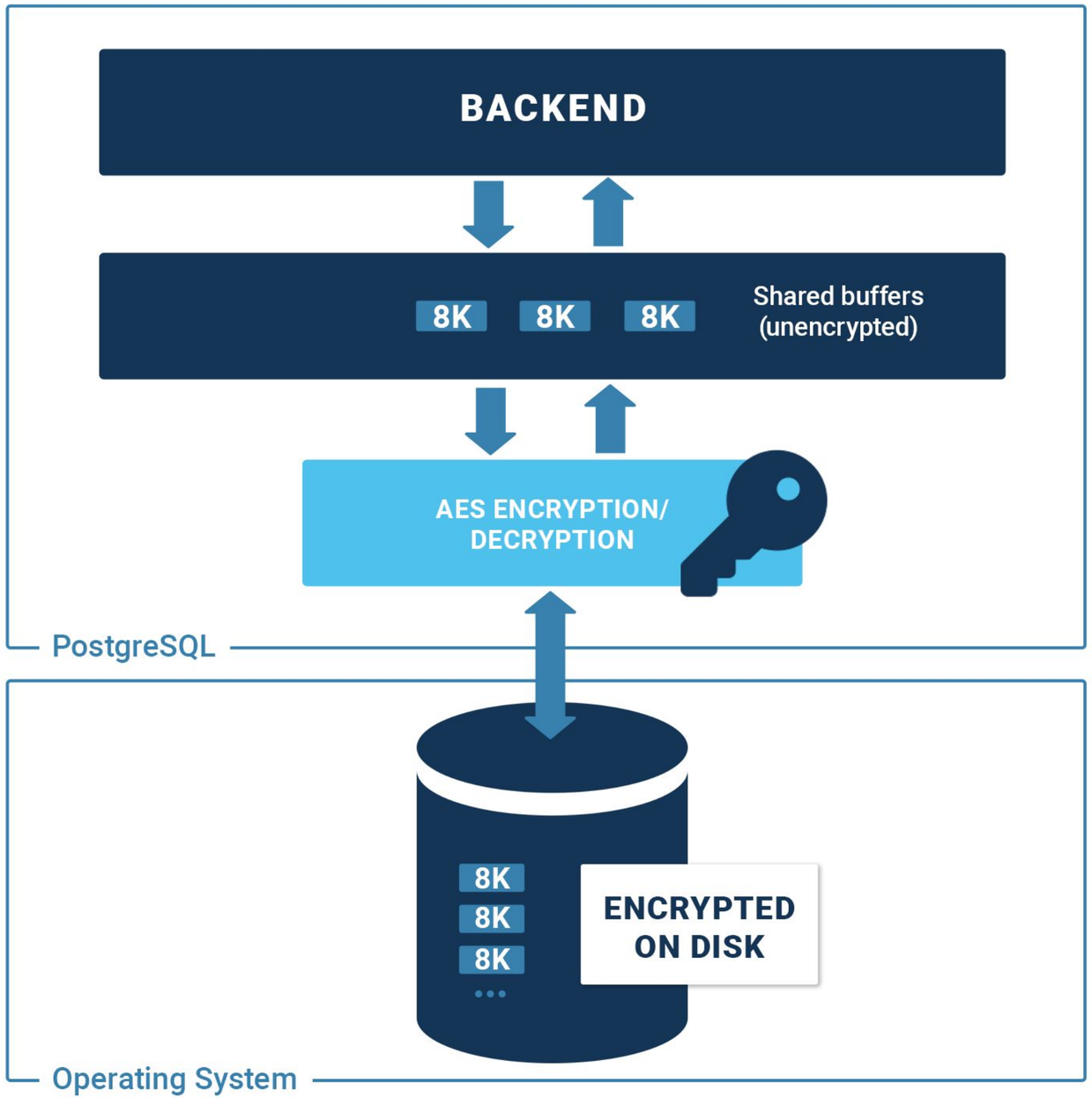
```
pci_dss=# \h ALTER DEFAULT PRIVILEGES
Command:     ALTER DEFAULT PRIVILEGES
Description: define default access privileges
Syntax:
ALTER DEFAULT PRIVILEGES
    [ FOR { ROLE | USER } target_role [, ...] ]
    [ IN SCHEMA schema_name [, ...] ]
    abbreviated_grant_or_revoke
where abbreviated_grant_or_revoke is one of:
GRANT { { SELECT | INSERT | UPDATE | DELETE |
   TRUNCATE | REFERENCES | TRIGGER | MAINTAIN }
    [, ...] | ALL [ PRIVILEGES ] }
    ON TABLES
    TO { [ GROUP ] role_name | PUBLIC } [, ...]
   [ WITH GRANT OPTION ]
```

HINT



Encryption & Storage

Data at rest



Transparent Data Encryption

CYBERTEC PGEE

```
postgres=# create table encrypted(t text);
CREATE TABLE
postgres=# insert into encrypted values ('this is a
test to check if the text is stored encrypted');
INSERT 0 1
```

```
postgres@ubuntupgee:~$ hexdump -C /var/lib/postgresql/16/main/base/5/16388
00000000 00 00 00 00 00 4f 54 01 00 00 00 00 1c 00 b0 1f |.....OT.........|
00000010 00 20 04 20 00 00 00 00 b0 9f a0 00 00 00 00 00 |. . ............|
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
*
00001fb0 e9 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00001fc0 01 00 01 00 02 08 18 00 71 74 68 69 73 20 69 73 |........this is|
00001fd0 20 61 20 74 65 73 74 20 74 6f 20 63 68 65 63 6b | a test to check|
00001fe0 20 69 66 20 74 68 65 20 74 65 78 74 20 69 73 20 | if the text is |
00001ff0 73 74 6f 72 65 64 20 65 6e 63 72 79 70 74 65 64 |stored encrypted|
00002000
postgres@ubuntupgee:~$
```

https://www.cybertec-postgresql.com/en/tde-a-dive-into-encrypted-data/

```
postgres=# create table encrypted(t text);
CREATE TABLE
postgres=# insert into encrypted values ('this is a test to
check if the text is stored encrypted');
```

```
postgres@ubuntupgee:~$ hexdump -C /var/lib/postgresql/16/pgee/base/5/16388
00000000 00 00 00 00 f0 e1 79 01 d3 6a bc 42 a5 95 e8 8b |......y..j.B....|
00000010 1e df 30 ad 5f 61 0f 55 6d af f4 79 9b ba 77 69 |..0._a.Um..y..wi|
00000020 c6 c7 c5 61 c5 4c f4 a8 00 64 35 af 79 fa e7 fd |...a.L...d5.y...|
00000030 09 a4 07 13 2e c1 76 45 1f ed 03 42 3d d5 1f 8f |......vE...B=...|
00000040 38 3f 87 ad 7c d0 47 79 ed 05 1b 18 34 b6 68 ff |8?..|.Gy....4.h.|
00000050 f6 65 63 4d e2 04 16 bd 3b 8a df 20 22 ff f3 7f |.ecM....;.. '...|
00000060 f1 93 6f 04 f3 e1 94 43 8d 19 e4 8b e9 4e 64 3e |..o....C.....Nd>|
00000070 4c c9 94 ab 64 da d1 d7 39 66 85 36 65 d1 0d 84 |L...d...9f.6e...|
00000080 e8 fd 7e 38 52 7e e3 39 20 4d eb 56 f9 e2 20 f6 |..~8R~.9 M.V.. .|
00000090 77 1e 83 ec 66 9d 8b 95 90 4a 0c 2a 72 ad 0f 24 |w...f....J.*r..$|
000000a0 5a 2f 9f d9 16 31 f5 6a b9 8e 88 18 c3 ed 3f 28 |Z/...1.j......?(|
000000b0 97 d6 27 6c df 37 68 f5 22 3b 0f 57 9e ee 3b b5 |..'l.7h.";.W..;.|
000000c0 39 b8 7a 32 7f a0 13 3d ea 94 1d be c4 84 27 74 |9.z2...=......'t|
000000d0 c1 28 93 c5 1e ee ff 56 bb 0f 91 47 57 38 a6 fb |.(.....V...GW8..|
000000e0 60 45 a1 d0 83 8b e8 45 e7 86 71 1d 44 b6 93 d3 |`E.....E..q.D...|
000000f0 b8 af e7 a7 85 47 d9 dc bf 78 60 02 74 8c 84 2c |.....G...x`.t..,|
00000100 e8 56 86 af d9 78 2f 9d c5 ff 3b 33 30 b7 5e bc |.V...x/...;30.^.|
00000110 d8 5d 83 c6 c3 50 91 68 e8 30 c6 82 eb be 07 36 |.]...P.h.0.....6|
00000120 8d 1c 53 a0 e8 5d 58 61 0a 90 7c 38 7f de 74 d4 |..S..]Xa..|8..t.|
00000130 03 e0 f9 ec 9a 85 97 b3 47 3c fa 15 74 0c af 49 |........G<..t..I|
00000140 1c b8 71 40 7a 17 c4 eb be a7 f1 46 6b d1 1f 57 |..q@z......Fk..W|
00000150 bd 5a f0 36 ac da c9 f5 ce 48 79 3a d1 7d 58 bb |.Z.6.....Hy:.}X.|
00000160 b3 f1 29 d8 0b 13 e8 8f 04 34 71 29 3d fd 1f 90 |..).......4q)=...|
```

https://www.cybertec-postgresql.com/en/tde-a-dive-into-encrypted-data/

safety from .....


WARNING:
Data Breach!
PostgreSQL
CYBERTEC
cybertec-postgresql.com

and what about the “DUMP” ?

```
compliance=# create extension pgcrypto;
CREATE EXTENSION


compliance=# INSERT INTO credit_cards (card_holder,
card_number_enc)
       VALUES ('Alice', pgp_sym_encrypt('4111111111111111',
     'mysecretkey'));
INSERT 0 1
compliance=# SELECT * from credit_cards;
 id | card_holder |                          card_number_enc
----+-------------+-----------------------------------------------------
  1 | Alice       | \xc30d0407030239caae4893c1920279d24101d722
```

```
compliance=# SELECT card_holder, pgp_sym_decrypt(card_number_enc,
'mysecretkey') AS card_number
FROM credit_cards;

 card_holder |    card_number
-------------+--------------------
 Alice       | 4111111111111111
(1 row)
```

let’s try something ….. log what ???

```
compliance=# SHOW log_statement;
 log_statement
---------------
 all
```

```
compliance=# INSERT INTO credit_cards (card_holder, card_number_enc)
VALUES ('Bob', pgp_sym_encrypt('5555444433332222', 'mysecretkey'));
```

```
2025-10-11 11:39:57.951 UTC [45] LOG:  statement: INSERT INTO
credit_cards (card_holder, card_number_enc) VALUES ('Bob',
pgp_sym_encrypt('5555444433332222', 'mysecretkey'));
```

What about pg_stat_activity ?

```
compliance=# select datid, datname, usename, application_name, query  from
pg_stat_activity;
\watch
 datid |  datname   | usename  | application_name |
query
-------+------------+----------+------------------+------------------------------
 16388 | compliance | postgres | psql | INSERT INTO credit_cards (card_holder,
card_number_enc) VALUES ('Bob', pgp_sym_encrypt('5555444433332222', 'mysecretkey'));
```

Row Level Security

```
ALTER TABLE table_name ENABLE ROW LEVEL SECURITY;
```

```
rls_lab=# select current_user;
 current_user
--------------
 postgres
(1 row)

rls_lab=# select * from payroll;
 id | employee | salary | username
----+----------+--------+----------
  1 | Alice    |  85000 | alice
  2 | Bob      |  83000 | bob
  3 | Eve      |  90000 | eve
(3 rows)

rls_lab=# SET ROLE alice;
SET
rls_lab=> select * from payroll;
 id | employee | salary | username
----+----------+--------+----------
  1 | Alice    |  85000 | alice
(1 row)
```

1. Create table
2. ENABLE ROW LEVEL SECURITY for the table.
   CREATE POLICY *policy_name* ON *table_name* TO *ROLE/USER*
   USING (condition <e.g. manager = current_user>);

https://www.postgresql.org/docs/current/ddl-rowsecurity.html

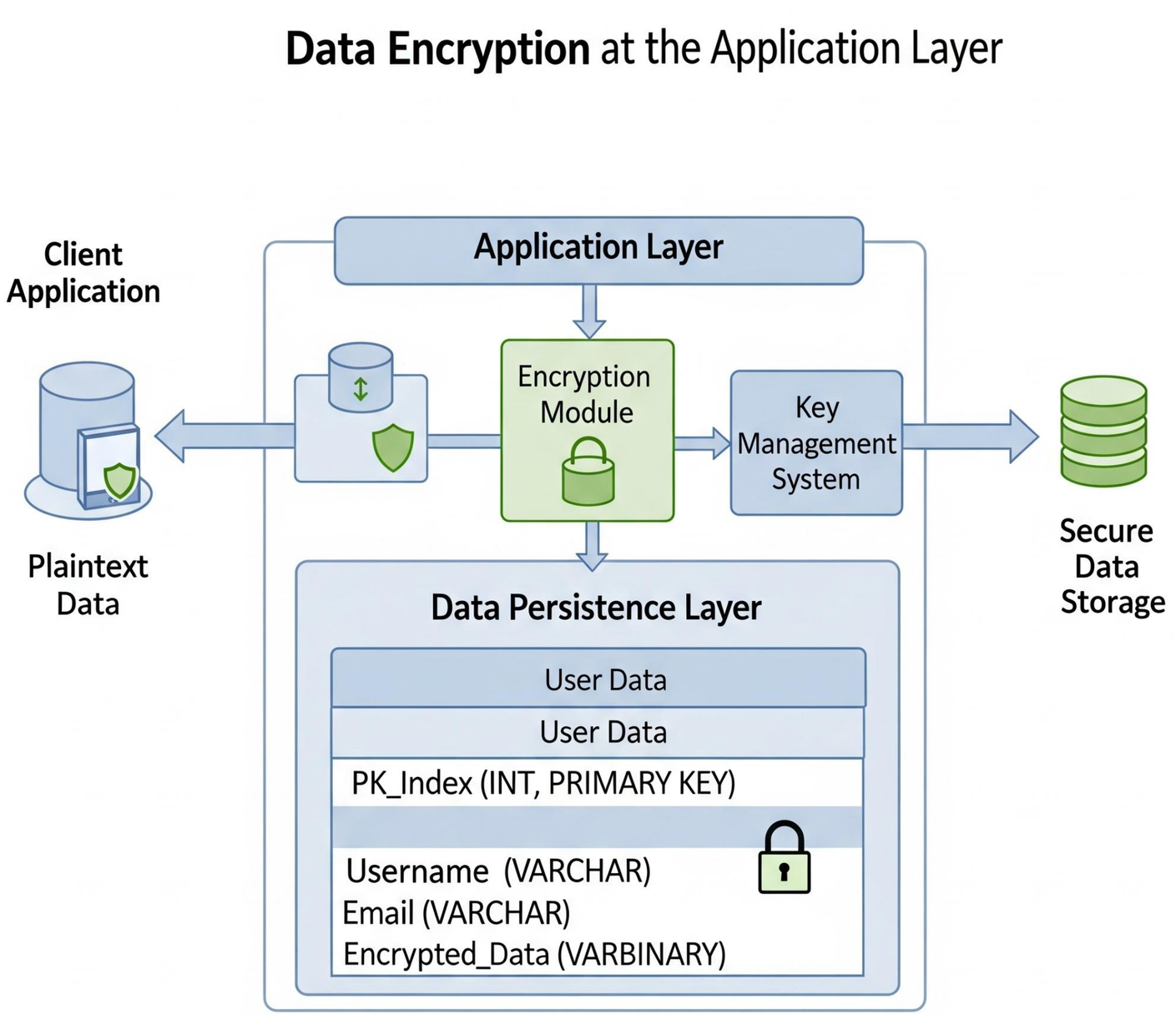
```
rls_lab=# SELECT rolname, rolsuper, rolbypassrls FROM pg_roles WHERE
rolsuper OR rolbypassrls;
```

```
 rolname  | rolsuper | rolbypassrls
----------+----------+--------------
 postgres | t        | t
(1 row)
```

ALTER TABLE ... FORCE ROW LEVEL SECURITY.

next step ...

1. Encrypt at APP layer.
2. Make sure to have an PK_index column.



PK_Index Primary Key Index for efficient data retrieval

Those last 3 digits! …

Never store the CVV number!

Network security

Data in Transit

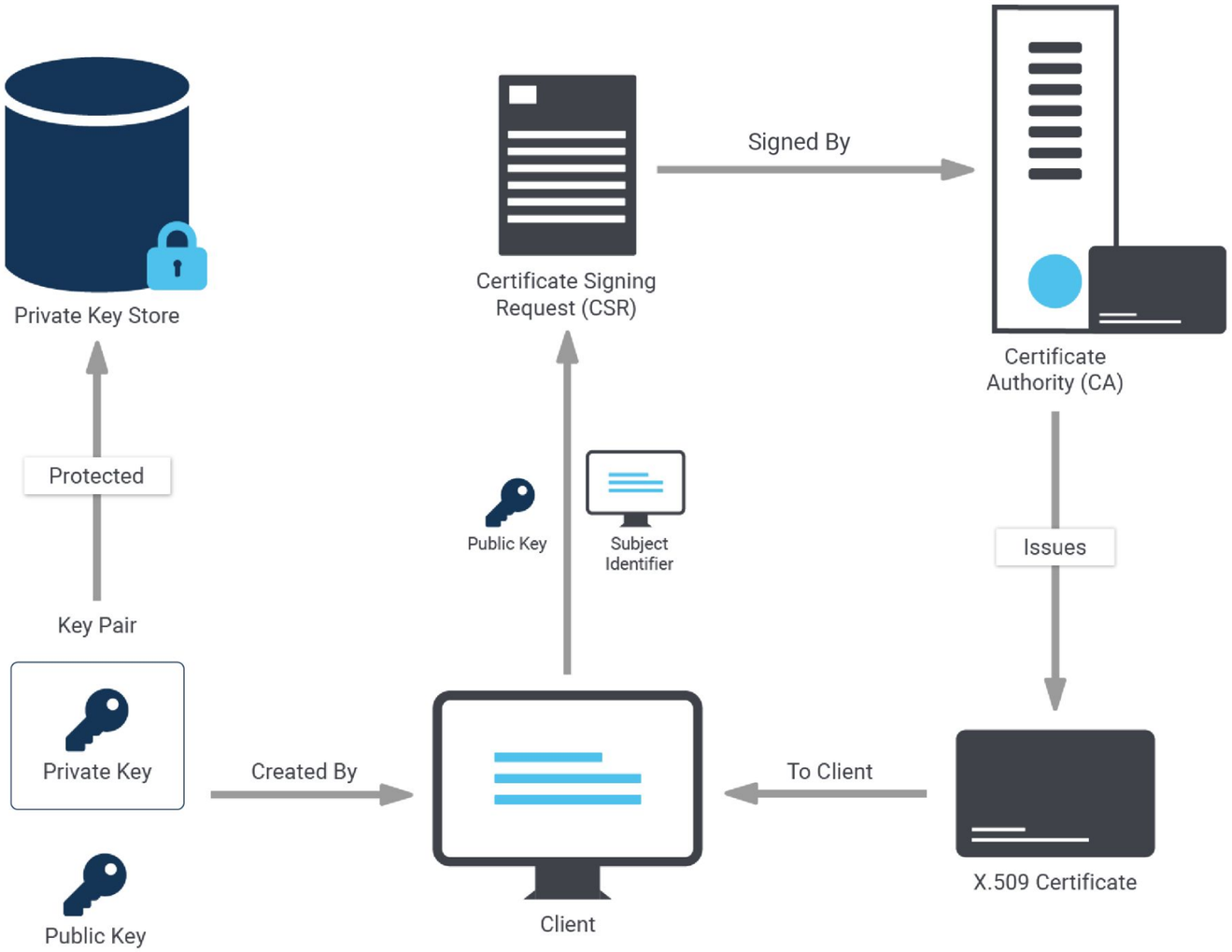
Network security

“Never talk to strangers”

- PostgreSQL provides SSL
  - Encrypt client / server connection
- Multiple SSL levels supported
  - Performance impact depends on SSL level
- Host based access control
  - Limit client IPs
  - Control authentication method (SSO, etc.)

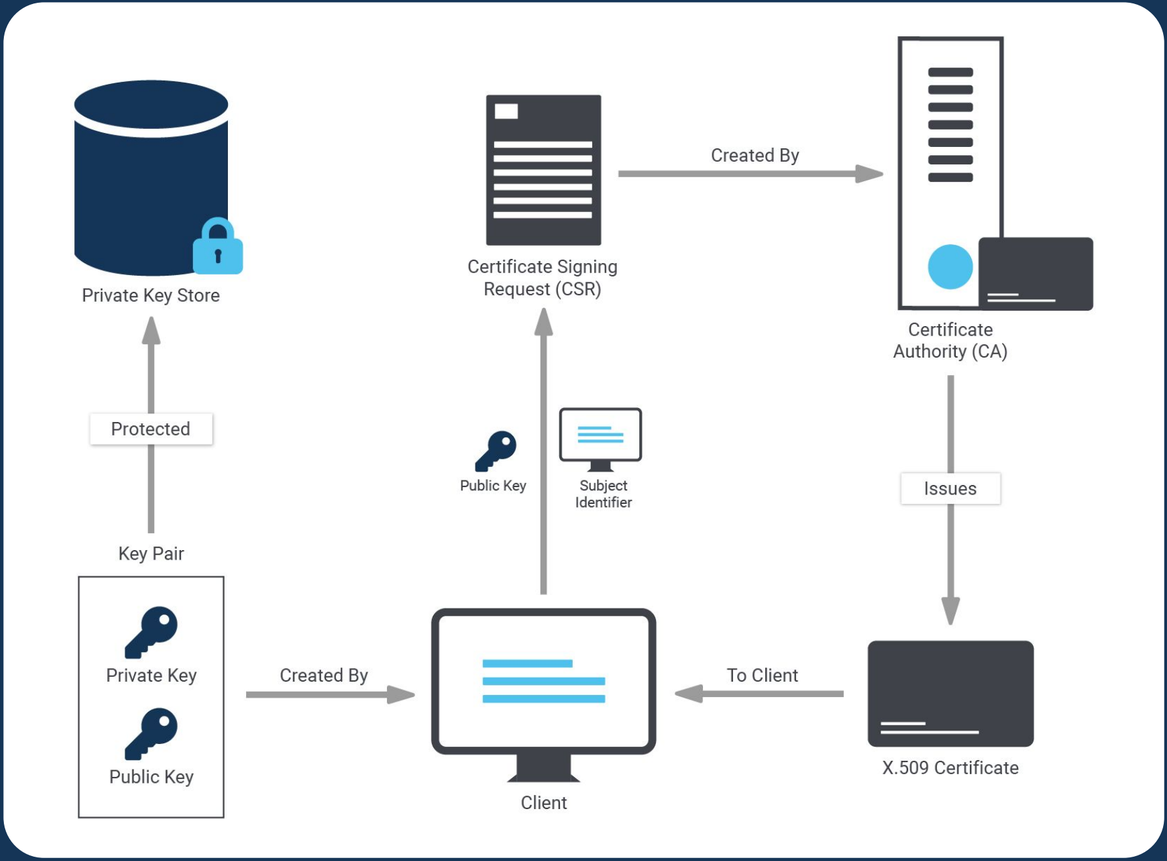
Secure TCP/IP Connections with SSL

“How SSL works”



Secure TCP/IP Connections with SSL

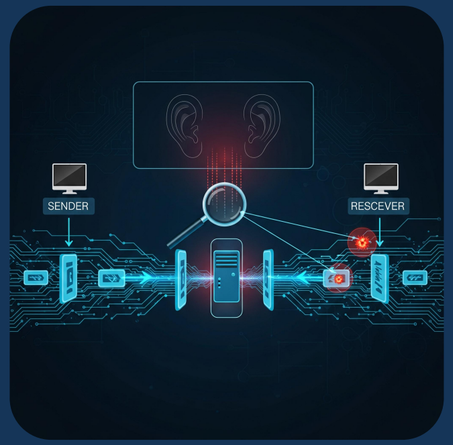
“How SSL works”



Eavesdropping

`sslmode=require`

`sslmode=verify-full`



MITM

SECURITY

PostgreSQL SSL Modes

sslmode	Eavesdropping protection	MITM protection	Statement
disable	No	No	I don't care about security, and I don't want to pay the overhead of encryption.
allow	Maybe	No	I don't care about security, but I will pay the overhead of encryption if the server insists on it.
prefer	Maybe	No	I don't care about encryption, but I wish to pay the overhead of encryption if the server supports it.
require	Yes	No	I want my data to be encrypted, and I accept the overhead. I trust that the network will make sure I always connect to the server I want
verify - ca	Yes	Depends on CA policy	I want my data encrypted, and I accept the overhead. I want to be sure that I connect to a server that I trust.
verify - full	Yes	Yes	I want my data encrypted, and I accept the overhead. I want to be sure that I connect to a server I trust, and that it's the one I specify.

Encryption Matters

- Different levels
- Decide on requirements

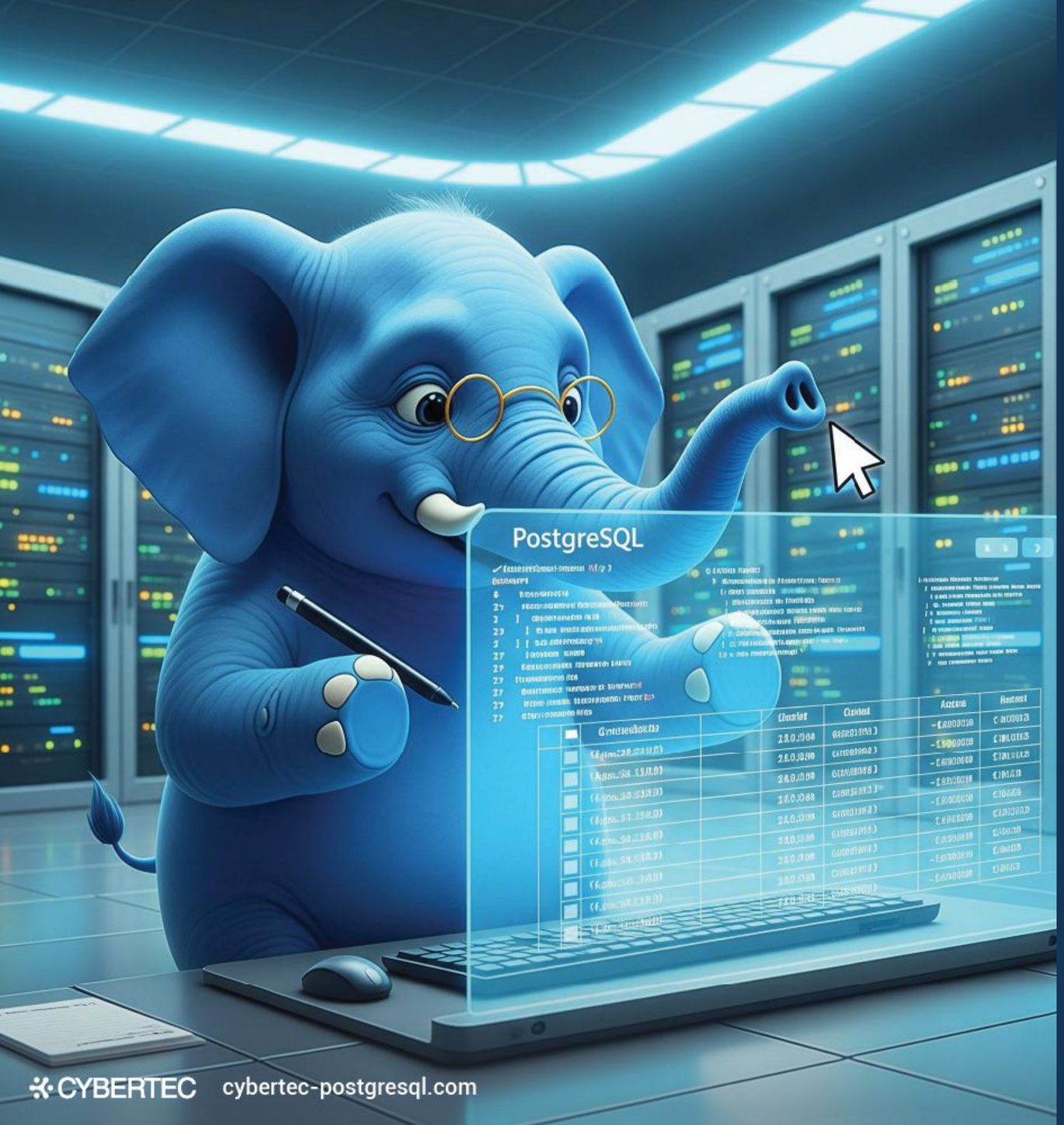
Ref link: https://www.postgresql.org/docs/current/libpq-ssl.html

Reduce Attack Surface

1. Place DB inside private network, restrict ingress
2. Use network firewalls, VPNs, DMZs
3. Use separate DB clusters for PCI/PHI vs general data
4. Use VLANs, subnets, zero-trust

Auditing Access

Track and document user activity



AUDIT

Auditing PostgreSQL

Keeping an eye on things

- Track DBA and user activity
  - Changes to database objects
  - Log critical activity

- Various solutions
  - pgaudit: Standard audit
  - pg_permissions
  - CYBERTEC audit: Extended functionality

- Define the “ideal world”
- Compare it to reality
- “Automate” compliance
- Enhance security

pg_permissions

Easy privilege audit

```
INSERT INTO public.permission_target
   (role_name, permissions,
    object_type, schema_name)
VALUES
   ('appuser', '{USAGE}',
    'SCHEMA', 'appschema');

INSERT INTO public.permission_target
   (role_name, permissions,
    object_type, schema_name, object_name)
VALUES
   ('appuser', '{USAGE}',
    'SEQUENCE', 'appschema', 'appseq');

SELECT * FROM public.permission_diffs();
 missing | role_name | object_type | schema_name | object_name | column_name | permission
---------+-----------+-------------+-------------+-------------+-------------+------------
 f       | laurenz   | VIEW        | appschema   | appview     |             | SELECT
 t       | appuser   | TABLE       | appschema   | apptable    |             | DELETE
(2 rows)
```

CYBERTEC Audit

one audit to rule them all....

- Collect audit via background worker
- Flexible output plugins
  - Parquet (highly compressed)
  - CSV (for easy reading)
  - Table output
  - Network Transfer(Transparent)
- Audit at scale beyond - pgaudit

Security Information & Events Management

wazuh!!

```
<decoder name="postgresql">
 <prematch type="pcre2">(?i)statement:</prematch>
</decoder>
<decoder name="postgresql_child">
 <parent>postgresql</parent>
 <regex
type="pcre2">(?i)\d+-\d+-\d+\s+\d+:\d+:\d+.\d+\s+\S+\s+\[
\d+\]\s+(\S+\s*\S+)@\S+\s+LOG:\s+statement:\s+create\s+da
tabase\s+([^"\s;]+)</regex>
 <order>db_user,database</order>
</decoder>
```

https://wazuh.com/blog/monitoring-postgresql-database-with-wazuh/

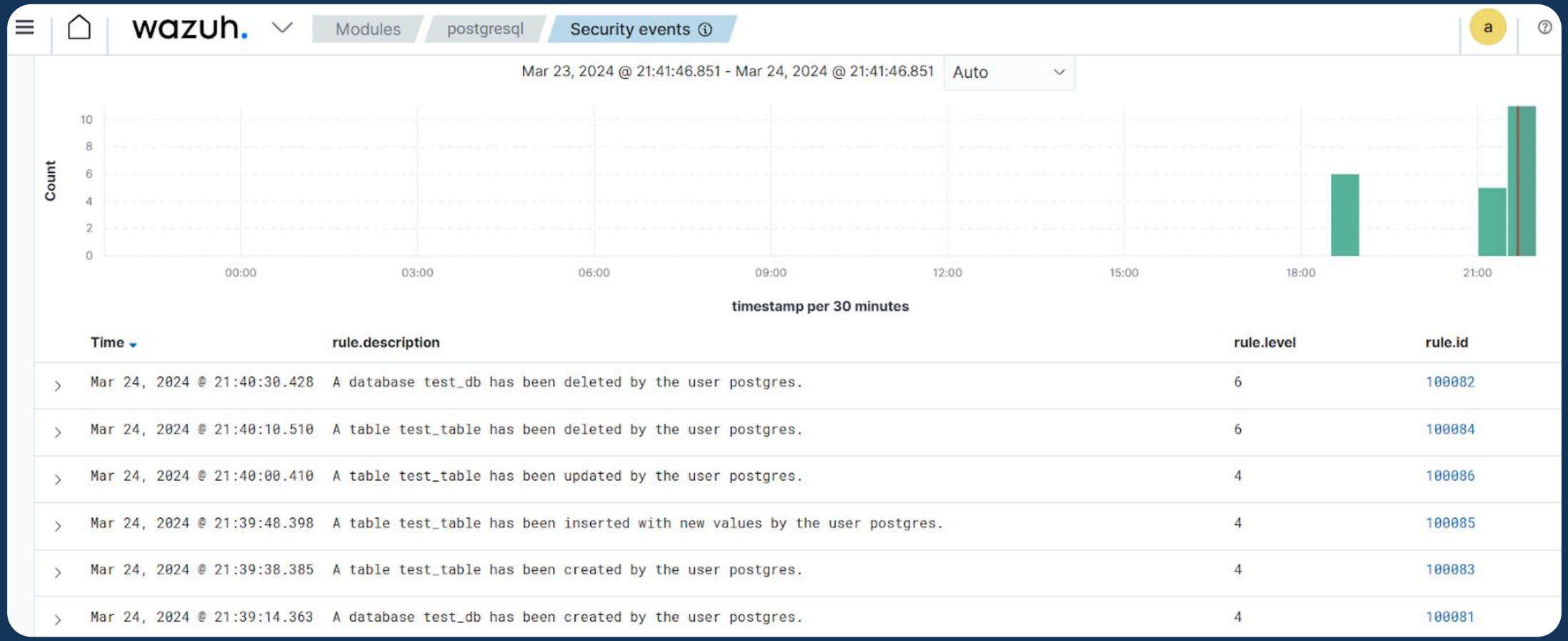
wazuh!!

```
<group name="postgresql,">
<!-- This rule detects when a database is created.-->
  <rule id="100081" level="4">
    <if_sid>100080</if_sid>
    <match type="pcre2">(?i)create database</match>
    <description>A database $(database) has been created by the user
$(db_user).</description>
  </rule>

<!-- This rule detects when a database is deleted.-->
  <rule id="100082" level="6">
    <if_sid>100080</if_sid>
    <match type="pcre2">(?i)drop database</match>
    <description>A database $(database) has been deleted by the user
$(db_user).</description>
    <mitre>
      <id>T1485</id>
    </mitre>
  </rule>
</group>
```

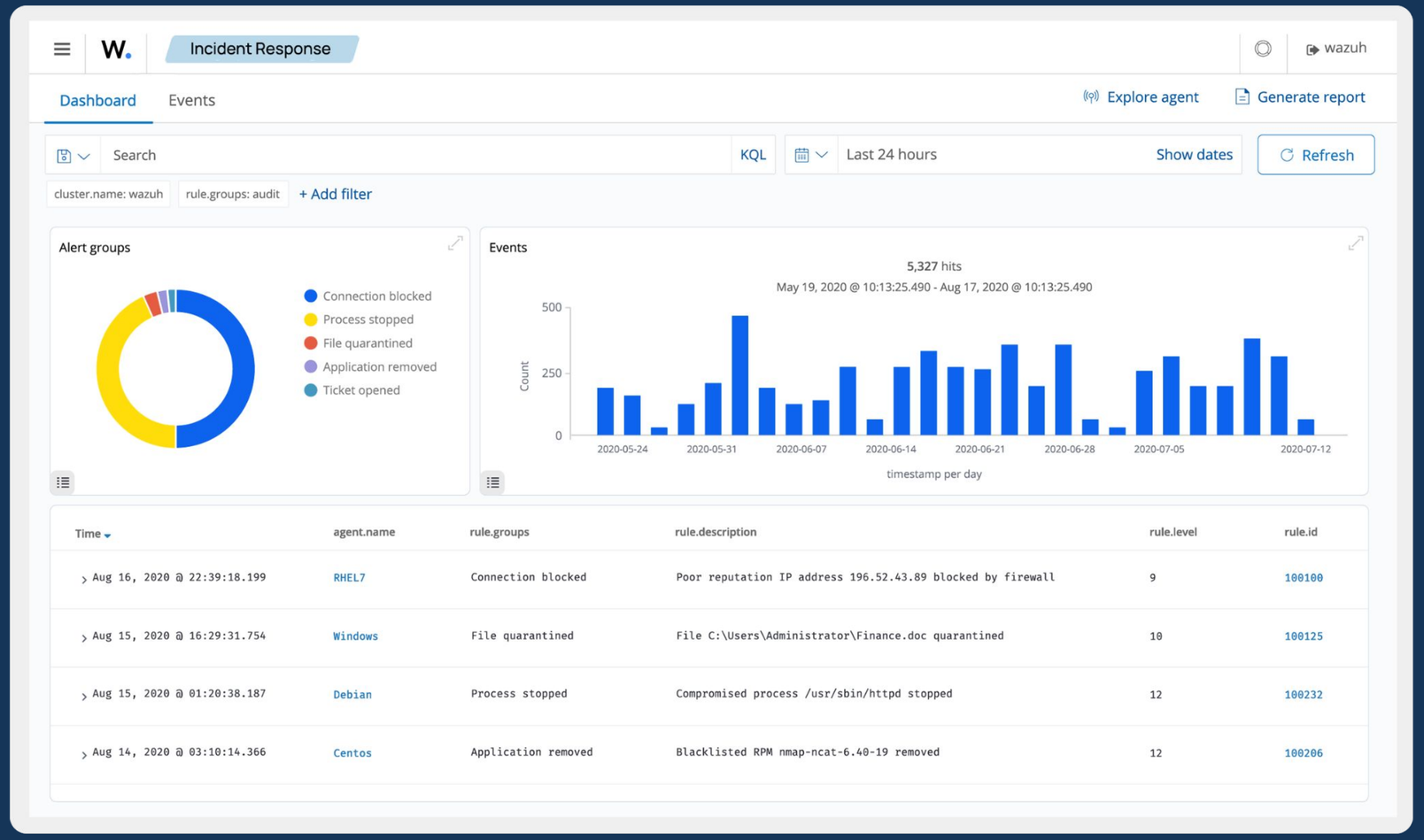
https://wazuh.com/blog/monitoring-postgresql-database-with-wazuh/

SIEM & Incident response



Time	rule.description	rule.level	rule.id
Mar 24, 2024 @ 21:40:30.428	A database test_db has been deleted by the user postgres.	6	100082
Mar 24, 2024 @ 21:40:10.510	A table test_table has been deleted by the user postgres.	6	100084
Mar 24, 2024 @ 21:40:00.410	A table test_table has been updated by the user postgres.	4	100086
Mar 24, 2024 @ 21:39:48.398	A table test_table has been inserted with new values by the user postgres.	4	100085
Mar 24, 2024 @ 21:39:38.385	A table test_table has been created by the user postgres.	4	100083
Mar 24, 2024 @ 21:39:14.363	A database test_db has been created by the user postgres.	4	100081

Incident Response



https://wazuh.com/

Security and Updates

Remediate vulnerabilities

PostgreSQL Updates Policy & HA

Frequency of new releases

- Minor release every quarter
- Clear Roadmap

Compatibility

- Minor releases are only bug fixes
- Safe to deploy

High-Availability

- Guaranteed Business Continuity & Uptime
- Streamlined DR
- Built-in Load Read Scalability

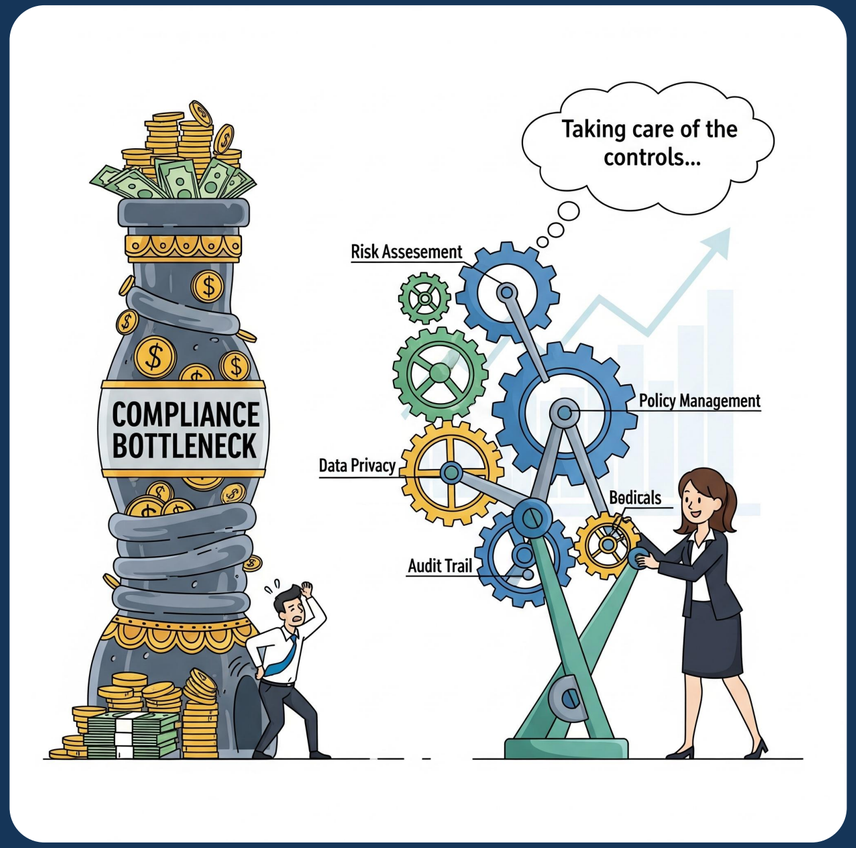
Deploying secure code

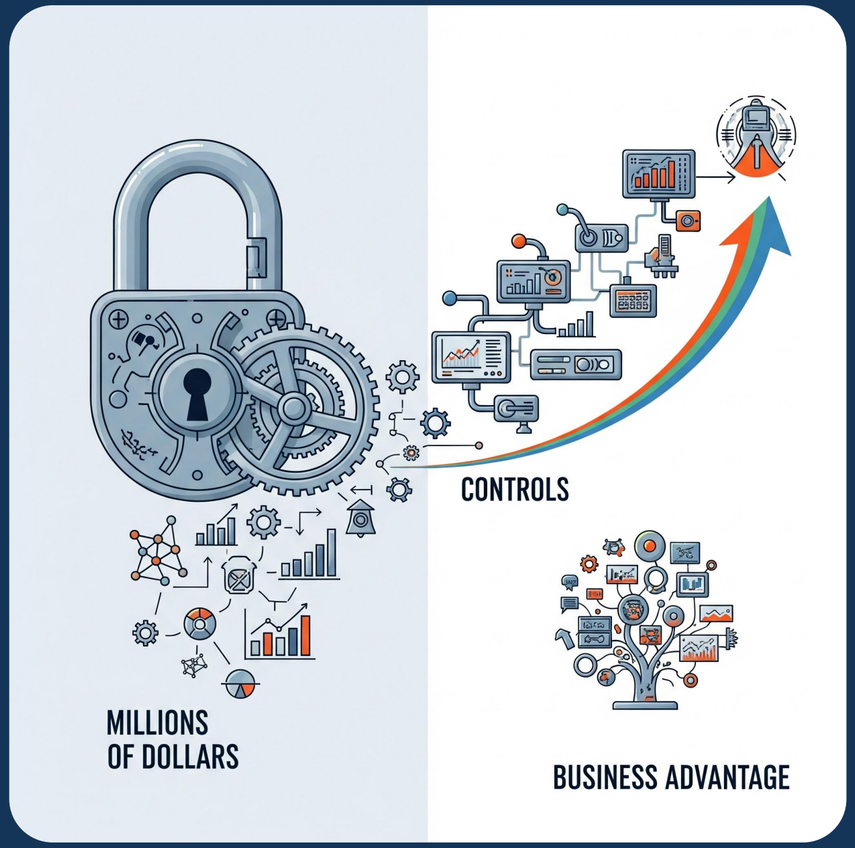
“Keeping binaries up to date”

- PostgreSQL provides quick updates
  - Install new binaries
  - Fast database restart
- **Audited build pipeline**
  - Keep an eye on CVEs
  - Document what is in the binaries
- PostgreSQL fixes are provided
  - When leaks are found
  - When data is at risk

Control Area	Common Best Practice (All)	GDPR	PCI DSS	HIPAA
Access Control	**Role-based access; no shared superuser; RLS for sensitive data**	Support Right to Access/Deletion; log data access by subject	Limit CHD access; strong auth + MFA	Unique user IDs; least privilege for PHI
Encryption	**TLS in transit; at-rest encryption via filesystem + pgcrypto**	Pseudonymize/anonymize personal data	Field-level PAN encryption or tokenization	Encrypt PHI at rest and in transit
Data Minimization	**Store only required attributes; define retention**	Legal basis mapping & retention enforcement	Limit stored PAN or use token service	Store minimum PHI necessary for treatment
Auditing & Monitoring	**Enable pgaudit, central SIEM integration**	Track personal data access & erasure events	Log all CHD access and admin actions	Monitor PHI access; tamper-evident logs
Backup & Recovery	**Encrypted backups; restore tested & audited**	Ensure deletion propagates to backups	Retain logs ≥1 year; backup security	DR plan for PHI availability & integrity
Data Subject / Individual Rights	**APIs or scripts for delete/export**	Implement Right to Erasure + Data Portability	N/A	Provide audit trails for patient data
Application Layer Controls	**App-side encryption; tokenization**	Consent capture, data lineage	Tokenization & masking	App-side key mgmt and integrity checks
Environment Segmentation	**Separate prod/test; network firewalls**	Limit data transfer outside EU	Isolate PCI zone (DB + app)	Controlled PHI environments (with BAA)
Key Management	**Use external KMS or HSM; rotate keys regularly**	Secure key vault for pseudonymized data	KMS/HSM required for PAN encryption	KMS/HSM for PHI encryption keys
Dev/Test Data Handling	**Synthetic or masked data only**	Remove all identifiers	No CHD in test	De-identify PHI in non-prod
Incident Response	**Logged, auditable, rehearsed plan**	Breach notification within 72 hrs	Report per PCI incident process	HIPAA breach notification (≤60 days)

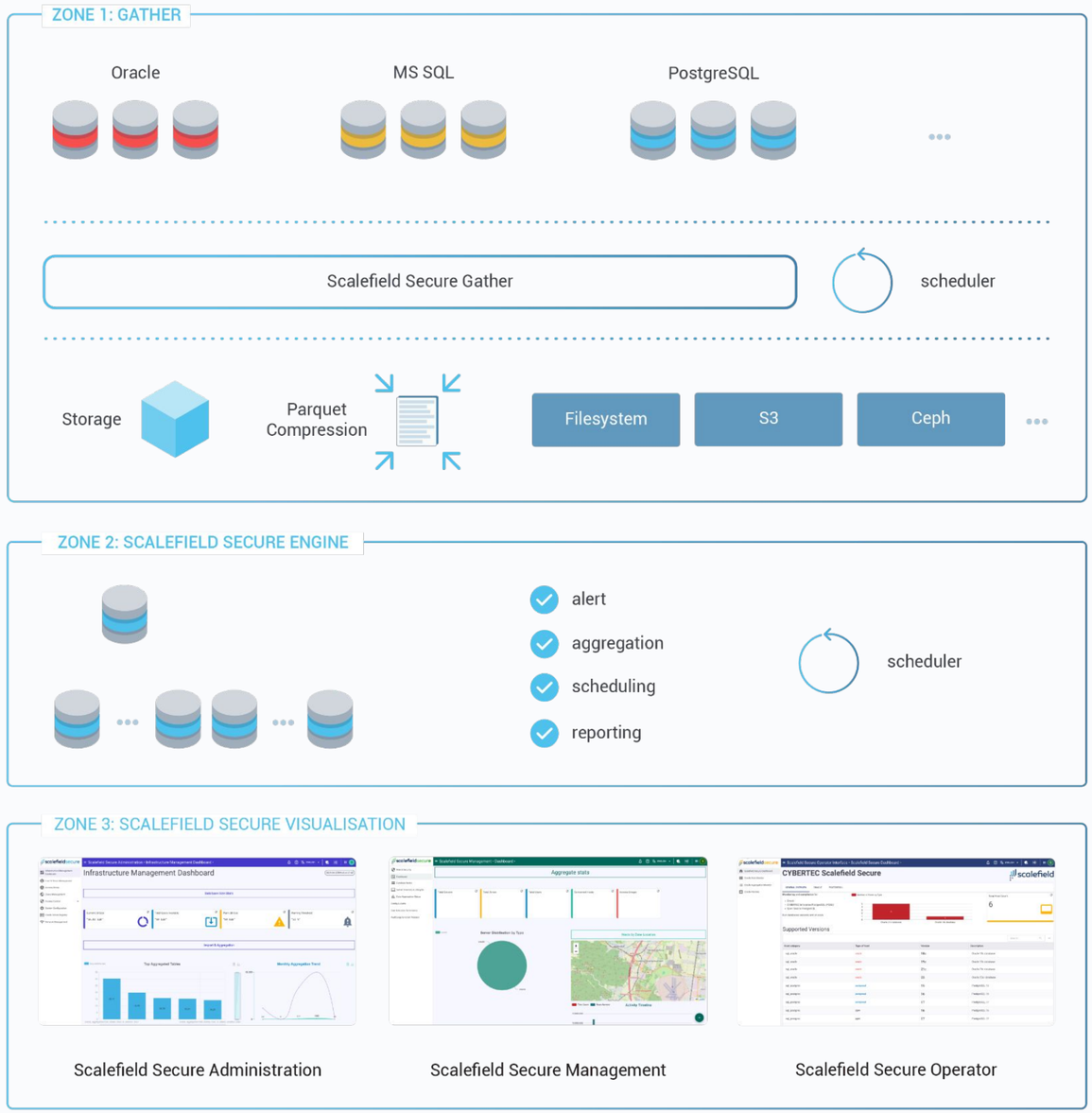
Turning compliance into a business advantage instead of a bottleneck





scalefieldsecure

- Compliance at a glance
  - Auditor Friendly
- Audit all important databases centrally.
  - Oracle, DB2 LUW, MS SQL
  - PostgreSQL, PGEE, EPAS



PostgreSQL

Empowering people

Let us make it happen

- The world should be governed by
- No vendor lock-ins
- Open Source
- Digital Independence
- Secure & Cost effective Infrastructure

Our partners at PGConf.EU

readyset

redgate

OpenSource^DB
Your Trusted Data Partner

Microsoft

PROVENTA

Open Alliance

For PostgreSQL Education

Your Pathway to Verified PostgreSQL Skills

Scan for Updates

oapg-edu.org

PGDay Austria returns in 2026

Scan for updates

pgday.at

Questions?