Best practices for Fujitsu Enterprise Postgres database security

FUJITSU

Contents at a glance

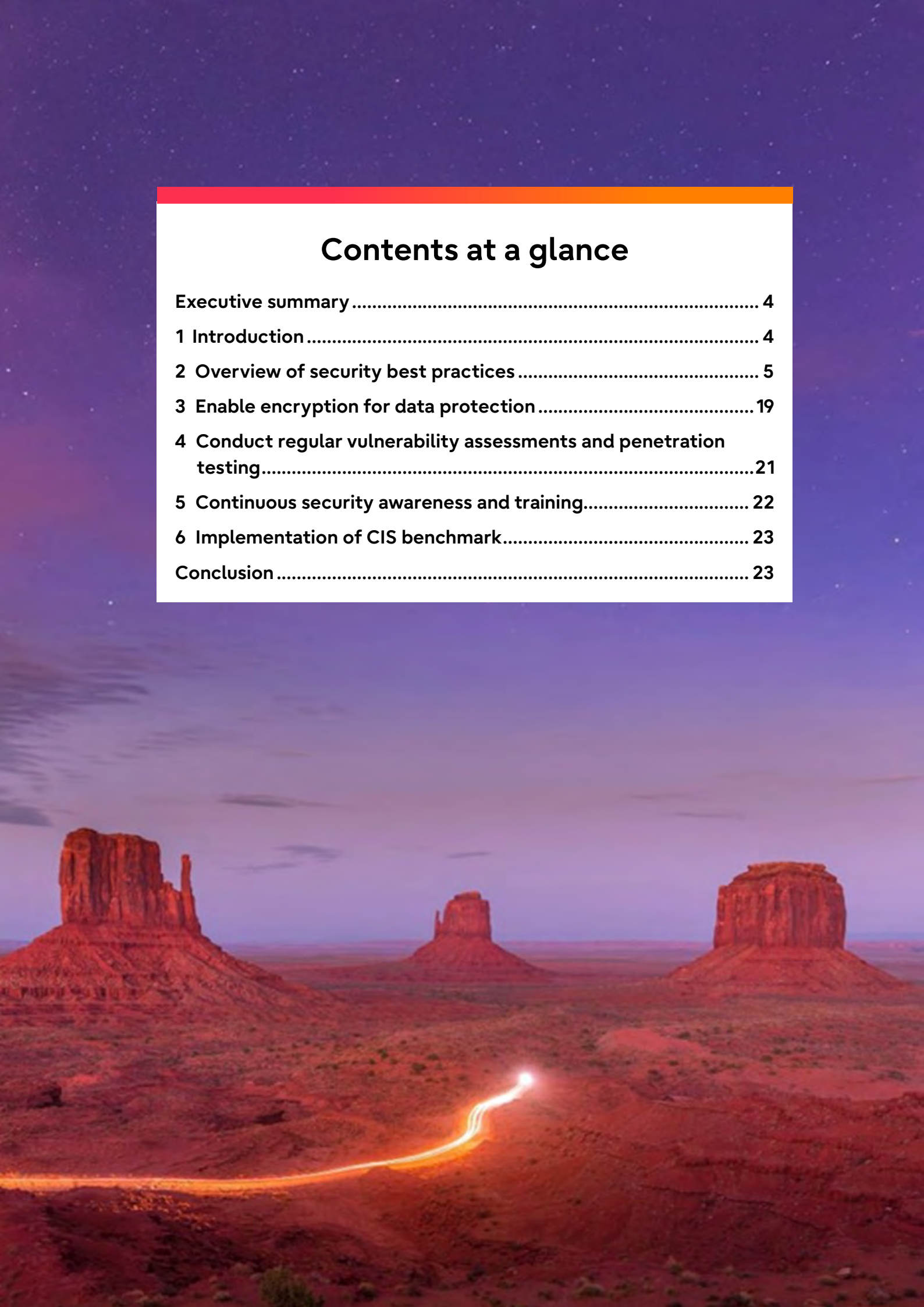


Table of contents

Executive summary

This paper offers structured and useful suggestions for PostgreSQL database security. It employs a multi-layered security approach that addresses topics like host-level access controls, safeguards for networks, database user management, and encryption methods. All these factors are important, but the emphasis is on PostgreSQL-specific techniques to enforce database and content security.

The document uses the AAA framework (Authentication, Authorization, and Auditing) to address database security methodically, which is a standard methodology in IT and network security.

The recommendations presented here apply to both the open-source PostgreSQL and **Fujitsu Enterprise Postgres** (FEP), which is a commercial version with enterprise features that address critical enterprise needs. Within the scope of this paper, Fujitsu Enterprise Postgres provides enhanced security capabilities not found in open-source PostgreSQL, including techniques such as Transparent Data Encryption (TDE), policy-based password management, detailed auditing, data masking, role-based access control (RBAC), KMIP-certified Cloud-based key management for TDE and FIPS 140-2 compliant crypto library that protects your data at rest & data in flight through robust cryptographic algorithms.

For further information on Fujitsu Enterprise Postgres refer to *Fujitsu Enterprise Postgres :Key features*

1. Introduction

In today's digital landscape, securing database systems is paramount to protecting sensitive information and maintaining the integrity of business operations. Fujitsu Enterprise Postgres is a robust, enterprise-grade database solution that offers advanced features and capabilities to meet the stringent security requirements of modern enterprises. This document outlines the best practices for securing Fujitsu Enterprise Postgres, focusing on a layered security model that encompasses physical security, network security, host access control, database access management, and data encryption.

1.1. The importance of database security

Database security is critical for several reasons:

- **Protection of sensitive data**: Databases often store confidential information such as personal data, financial records, and intellectual property. Unauthorized access to this data can lead to severe consequences, including financial loss, legal penalties, and reputational damage.
- **Compliance with regulations**: Many industries are subject to regulatory requirements that mandate stringent security measures for data protection. Adhering to these regulations is essential to avoid legal repercussions and maintain customer trust.
- **Prevention of data breaches**: Data breaches can result from various threats, including cyberattacks, insider threats, and human error. Implementing robust security practices helps mitigate these risks and safeguard the database environment.

2. Overview of security best practices

The security best practices for Fujitsu Enterprise Postgres are organized following the well-established AAA framework: Authentication, Authorization, and Accounting. This structured approach ensures a comprehensive and layered defence mechanism for securing database environments. By adhering to this framework, organizations can achieve robust security that includes verifying user identities, controlling their actions, and maintaining detailed records of all activities for auditing and compliance purposes.

2.1. General recommendations for security

2.1.1. Avoid use of default values

- **Default port**: Change the default port (27500/5432) to a non-standard port to reduce the likelihood of automated attacks targeting Fujitsu Enterprise Postgres instances. Update the postgresql.conf file to reflect this change.
- **Superuser name**: Avoid using the default superuser name (fepuser/postgres). Instead, specify a unique and non-default name for the superuser during the database initialization (initdb).

2.1.2. Restrict database access

- Control connection sources
  - Use the pg_hba.conf file effectively to define which users can connect from which hosts and using which authentication methods. Ensure these configurations are as restrictive as possible.
  - Avoid using wide-open configurations such as 0.0.0.0/0 unless absolutely necessary and well-secured.
  - Refrain from using the trust authentication method, as it allows connections without passwords. Proceed for stronger authentication mechanisms like scram-sha-256, LDAP or certificate-based authentication.

2.1.3. Enforce principle of least privilege

- **Application users**: Ensure that application database users only have the minimum permissions necessary to perform their required tasks. They should never have superuser privileges.
- **DBA accounts**: Allow DBAs to use their own unique user IDs for database management tasks. For example, a DBA named Joy should use a user account named joy.
- **Access control**: Use GRANT and REVOKE commands to fine-tune user access to tables, schemas, and other database objects, ensuring that permissions align with job responsibilities.

2.1.4. Keep software updated

Regularly update both the operating system and Fujitsu Enterprise Postgres database to apply the latest security patches. Fujitsu supports notify and delivers all the updates and security fixes to address the known vulnerabilities and improve overall security. Having an adequate policy to keep up with the newest versions and install patches and upgrades as soon as possible which helps the organization to safeguard your database against from any cyberattacks.

2.1.5. Configure secure access controls

Regularly review and refine user roles, privileges, and schema permissions. Avoid granting privileges to PUBLIC unless necessary.

2.2. Authentication

Verifying the identity of users accessing the database is the first line of defense. Fujitsu Enterprise Postgres supports various authentication methods, including password profiles, SCRAM, LDAP, certificate-based authentication, and many more. Implementing strong authentication mechanisms helps prevent unauthorized access.

2.2.1. Implement robust authentication mechanisms,

Define strict access control policies helps regulate and restrict incoming and outgoing network traffic to the database server and also ensures that only allow connections from authorized sources and explicitly deny all other requests. Fujitsu Enterprise Postgres also supports the wide varieties of client authentication methods.

Generally, *authentication* is the process by which the database server establishes the identity of the client, and determines whether the service account (or the user who runs the client connection) is permitted to connect with the database username that was requested and into the database specified (explicitly or implicitly).

Right password authentication methods prevent you from being a victim of password sniffing attacks on untrusted connections. Here are some critical practices to put in place:

- **Authentication methods**: The pg_hba.conf file in PostgreSQL controls access to the database by specifying which users can connect, which databases they can access, and from which IP addresses they can connect (if using TCP/IP). This file also defines the authentication methods to be used. The choice of authentication method depends on your specific needs and use case.
  - **SCRAM-SHA-256**: SCRAM-SHA-256 (Salted Challenge Response Authentication Mechanism) is a secure authentication method used in Fujitsu Enterprise Postgres to enhance password security. It employs the SHA-256 hashing algorithm, which is more secure than older methods like MD5. SCRAM-SHA-256 uses a challenge-response protocol, where the server sends a challenge to the client, and the client responds with a hashed version of the password combined with the challenge. This ensures that the actual password is never transmitted over the network, reducing the risk of password sniffing.

    Passwords are stored on the server in a cryptographically hashed form, making it difficult for attackers to retrieve the original passwords even if they gain access to the hashed values. This method also protects against replay attacks, as each authentication attempt involves a new challenge. To use SCRAM-SHA-256, both the server and client must support this authentication method, which may require updating older client libraries.

    To enable SCRAM-SHA-256, set the `password_encryption` parameter to `scram-sha-256` in the postgresql.conf file and update the pg_hba.conf file to specify scram-sha-256 as the authentication method. This method is recommended for environments where security is a high priority, providing robust protection against various types of attacks.

- **MD5**: MD5 authentication in Fujitsu Enterprise Postgres is a method used to verify the identity of database users through a challenge-response mechanism. When a user attempts to connect, the server sends a random challenge, and the client responds with an MD5 hash of the password combined with the challenge. This process ensures that the actual password is never transmitted over the network, reducing the risk of password sniffing.

  The MD5 method stores passwords in the database as an MD5 hash, which is a combination of the username and password. While this method prevents passwords from being stored in plain text, it does not provide protection if an attacker manages to steal the password hash from the server. Additionally, the MD5 algorithm is considered less secure by modern standards due to vulnerabilities that can be exploited by determined attackers.

  To configure MD5 authentication, you need to specify md5 in the pg_hba.conf file for the relevant connections. Despite its limitations, MD5 is still widely used because it is simple to implement and provides a basic level of security. However, for enhanced security, it is recommended to use more secure methods like SCRAM-SHA-256.

- **CERT**: Certificate authentication in Fujitsu Enterprise Postgres uses SSL client certificates to verify the identity of users connecting to the database. This method requires both the client and server to have SSL enabled, ensuring that all communications are encrypted. When a client attempts to connect, the server requests a certificate, which must be signed by a trusted Certificate Authority (CA). The server then verifies the certificate against its list of trusted CAs.

  The Common Name (CN) attribute of the certificate is compared to the database username. If they match, the user is authenticated without needing a password. This method enhances security by ensuring that only clients with valid certificates can access the database, reducing the risk of unauthorized access.

  To set up certificate authentication, you need to configure the `ssl_ca_file` parameter in the postgresql.conf file to point to the CA certificates. Additionally, you must update the pg_hba.conf file to include `clientcert=verify-ca` or `clientcert=verify-full` for the relevant connections. This setup ensures that the server not only verifies the certificate chain but also checks the CN against the database username for a more secure authentication process.

- **PAM**: PAM (Pluggable Authentication Modules) authentication in Fujitsu Enterprise Postgres allows the database to leverage the host system's PAM framework for user authentication. This method operates similarly to password authentication but uses PAM as the underlying mechanism. When a user attempts to connect, Fujitsu Enterprise Postgres passes the username and password to the PAM service, which then validates the credentials against the configured authentication modules.

  The PAM service is located at `/etc/pam.d/`. In OSS PostgreSQL, postgresql is the service name; however, in Fujitsu Enterprise Postgres, you must create with a meaningful name, such as `fep` and it can be customized if needed. PAM can also optionally validate the remote host name or IP address of the connecting client. One key requirement is that the user must already exist in the Fujitsu Enterprise Postgres database before PAM can be used for authentication.

  To configure PAM authentication, you need to add the appropriate entries in the pg_hba.conf file, specifying `pam` as the authentication method and optionally setting the `pamservice` parameter. This setup allows Fujitsu Enterprise Postgres to integrate with various authentication systems supported by PAM, such as LDAP, Kerberos, or local system accounts, providing a flexible and secure authentication solution.

For more information about the PAM authentication with Fujitsu Enterprise Postgres, refer to *How to configure PAM authentication in Fujitsu Enterprise Postgres*.

- **LDAP**: LDAP (Lightweight Directory Access Protocol) authentication in Fujitsu Enterprise Postgres allows the database to authenticate users against an existing LDAP directory, such as Active Directory or OpenLDAP. This method centralizes user management, enabling administrators to control database access through the LDAP server, which simplifies user provisioning and de-provisioning.

  When a user attempts to connect, Fujitsu Enterprise Postgres uses the credentials provided to bind to the LDAP server. If the credentials are valid, the user is authenticated. LDAP authentication can operate in two modes: simple bind and search+bind. In simple bind mode, Fujitsu Enterprise Postgres constructs a distinguished name (DN) using a prefix and suffix around the username and attempts to bind with this DN. In search+bind mode, Fujitsu Enterprise Postgres first binds to the LDAP server with a fixed user and searches for the user's DN, then re-binds as the user to verify the credentials.

  To configure LDAP authentication, you need to modify the pg_hba.conf file to specify `ldap` as the authentication method and provide necessary parameters like the LDAP server address, port, and search base. This setup allows Fujitsu Enterprise Postgres to leverage existing LDAP security policies, such as password complexity and multi-factor authentication, enhancing overall security.

  For more information about the integration of LDAP with AD, refer to *Connecting Fujitsu Enterprise Postgres to Active Directory for Authentication using LDAP*.

- **RADIUS**: RADIUS (Remote Authentication Dial-In User Service) authentication in Fujitsu Enterprise Postgres allows the database to authenticate users against a RADIUS server. This method is similar to password authentication but uses RADIUS to verify the username and password. When a user attempts to connect, Fujitsu Enterprise Postgres sends an Access Request message to the configured RADIUS server, which includes the username, encrypted password, and NAS Identifier. The RADIUS server then responds with either an Access Accept or Access Reject message.

  For RADIUS authentication to work, the user must already exist in the Fujitsu Enterprise Postgres database. This method centralizes authentication, making it easier to manage user credentials and enforce security policies across multiple systems. You can configure multiple RADIUS servers for redundancy, and Fujitsu Enterprise Postgres will try them sequentially if the primary server does not respond.

  To set up RADIUS authentication, you need to specify the RADIUS server addresses, shared secrets, and optional parameters like port numbers and NAS Identifiers in the pg_hba.conf file. This setup enhances security by leveraging the robust authentication mechanisms provided by RADIUS, ensuring that user credentials are validated securely.

- **GSSAPI**: GSSAPI (Generic Security Services Application Program Interface) authentication in Fujitsu Enterprise Postgres allows for secure, single sign-on (SSO) authentication using Kerberos. This method is particularly useful in environments where centralized authentication is required, such as in large organizations with many users. When a user attempts to connect to the Fujitsu Enterprise Postgres database, GSSAPI uses Kerberos tickets to authenticate the user without transmitting passwords over the network.

The process involves the client obtaining a Kerberos ticket from the Key Distribution Center (KDC) and presenting it to the Fujitsu Enterprise Postgres server. The server then verifies the ticket against its own Kerberos keytab file. If the ticket is valid, the user is authenticated. This method ensures that both the client and server identities are verified, providing a high level of security.

GSSAPI also supports encryption of the communication channel, ensuring that data transmitted between the client and server is protected. To enable GSSAPI authentication, Fujitsu Enterprise Postgres must be built with GSSAPI support, and the appropriate configuration must be set in the pg_hba.conf file. Additionally, the server's keytab file must be properly configured and secured.

This method is ideal for environments that already use Kerberos for authentication, as it integrates seamlessly with existing infrastructure and provides robust security features.

- **SSPI**: SSPI (Security Support Provider Interface) authentication in Fujitsu Enterprise Postgres is a Windows-specific technology that enables secure, single sign-on (SSO) authentication. It is particularly useful in environments where both the client and server are running on Windows. SSPI operates in negotiate mode, which means it will use Kerberos for authentication when possible and automatically fall back to NTLM if Kerberos is not available.

  When a user attempts to connect to the Fujitsu Enterprise Postgres database, SSPI uses the credentials of the currently logged-in Windows user, eliminating the need to enter a separate username and password. This seamless integration with Windows authentication simplifies the login process and enhances security by leveraging existing Windows security policies.

  SSPI and GSSAPI (Generic Security Services Application Program Interface) can interoperate, allowing an SSPI client to authenticate to a GSSAPI server and vice versa. This interoperability is beneficial in mixed environments where both Windows and non-Windows systems are used.

  To configure SSPI authentication, you need to update the pg_hba.conf file to specify `sspi` as the authentication method for the relevant connections. Additionally, ensure that the Fujitsu Enterprise Postgres server is running under a Windows domain account if you are using domain-based authentication.

  SSPI is recommended for Windows environments due to its ease of use and robust security features, providing a reliable method for authenticating Fujitsu Enterprise Postgres users.

2.2.2. Enforce password policies,

Cybercriminals often use brute force attacks to guess passwords, and if they're able to crack your password, they may be able to gain access to sensitive information. If they manage to gain access to this information, they damage your company reputation and data. So, to establish the strict password policies require complex passwords with a combination of alphanumeric characters, special symbols, and minimum length. Additionally, enforce regular password changes to minimize the risk of password-related vulnerabilities.

From Fujitsu Enterprise Postgres 15 SP1 onwards, Fujitsu Enterprise Postgres implements a robust password policy to enhance database security. This feature allows the DBAs to define comprehensive login security policies through profiles. These profiles can be assigned to database users to enforce various password authentication policies and manage user access effectively. These policies enable effective login and password management according to the organization's security policy.

Key settings within these profiles include:

- **Managing dormant users**: Automatically lock users who have not connected to the database for an extended period, ensuring inactive accounts do not become security risks.
- **Password policies**: Enforce rules for password authentication, such as setting a password lifetime to ensure regular password changes and restricting password reuse to prevent recycling of old passwords.
- **Account lockout**: Lock user accounts after a certain number of failed login attempts to mitigate the risk of brute-force attacks. You can also set the period during which the lock will be automatically lifted.
- **Grace periods**: Define a grace period after the password expires, allowing users to continue operating the database temporarily. Additionally, set a gradual password rollover time for using old passwords after a change.

Key benefits,

- **Enhanced security**: Strong password policies significantly reduce the likelihood of password-related security breaches.
- **Compliance**: Many regulatory standards, such as GDPR and HIPAA, require organizations to implement robust password policies.
- **User education**: Enforcing these policies helps educate users about secure password practices.
- **Policy replication**: The password lifetime and lock state are shared across all servers in a replication setup. This means a user is locked based on their policy not only on the primary server but also for consecutive failed logins to the standby server.

Benefits of combining Fujitsu Enterprise Postgres password policy with PostgreSQL passwordcheck module

The passwordcheck module in PostgreSQL can further enhance the security provided by Fujitsu Enterprise Postgres password policies. Here are the key features and benefits:

- **Password complexity enforcement**: Enforces rules like minimum length, use of special characters, and prohibiting passwords similar to usernames, ensuring strong passwords.
- **Customization**: Allows database administrators to define and enforce custom password policies tailored to specific security needs.
- **Integration with PostgreSQL**: Operates at the database level and is triggered automatically during password changes (e.g., via `CREATE ROLE` or `ALTER ROLE` commands), ensuring consistent enforcement of password policies.
- **Prevention of weak passwords**: Prevents setting weak passwords that could compromise database security, enhancing overall protection.

By combining Fujitsu Enterprise Postgres password policies with the passwordcheck module, organizations can achieve a robust and comprehensive approach to password management, significantly enhancing database security and compliance.

For more information about this, refer to *Policy :based login security : level up your database security and access control*.

2.2.3. SSL#TLS encryption,

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) encryption are critical features in Fujitsu Enterprise Postgres, designed to ensure secure communication between the database server and client applications. These protocols provide an encrypted channel, safeguarding data in transit from eavesdropping, tampering, and unauthorized access. When SSL/TLS is enabled, all data exchanged between the client and the Fujitsu Enterprise Postgres server is encrypted, including sensitive information like authentication credentials and query results. This is particularly important when the database is accessed over untrusted networks, such as the internet or shared organizational infrastructures.

Key Features of SSL#TLS in PostgreSQL:

- **Encryption**: Ensures confidentiality by encrypting the data exchanged between the client and server.
- **Authentication**: Enables mutual authentication using SSL/TLS certificates, verifying both the server and, optionally, the client.
- **Data integrity**: Protects the data in transit from being modified or corrupted during transmission.

Fujitsu Enterprise Postgres supports several SSL/TLS configurations, allowing administrators to fine-tune the level of security based on their requirements. For instance:

- The server can be configured to accept SSL/TLS connections by setting the ssl parameter to on.
- Certificates, including server certificates, client certificates, and Certificate Authorities (CA), can be used for secure and mutual verification.

To enable SSL/TLS, the server requires a valid certificate and private key, while the clients must be configured to trust the server's certificate. Fujitsu Enterprise Postgres provides options for enforcing security, such as the `sslmode` parameter, which can be set to values like require, `verify-ca`, or `verify-full` to control how strictly the client validates the server's identity.

2.3. Authorization

Once authenticated, users must be granted appropriate permission to access and manipulate data. Fujitsu Enterprise Postgres allows fine-grained access control through role-based permissions, ensuring that users have the minimum necessary access to perform their tasks. This principle of the least privilege minimizes the risk of accidental or malicious data manipulation.

2.3.1. Assigned privileges

Privileges in Fujitsu Enterprise Postgres are granted to roles (users or groups) to perform specific operations on database resources like tables, schemas, databases, and more. These operations include SELECT, INSERT, UPDATE, DELETE, TRUNCATE, REFERENCES, and USAGE, among others.

Privileges can be granted or revoked using SQL commands such as GRANT and REVOKE. Roles can be defined as login roles (which can authenticate) or non-login roles (used primarily for grouping privileges). The role-based access control (RBAC) system allows administrators to group privileges into roles and assign those roles to users, ensuring modular and manageable access control.

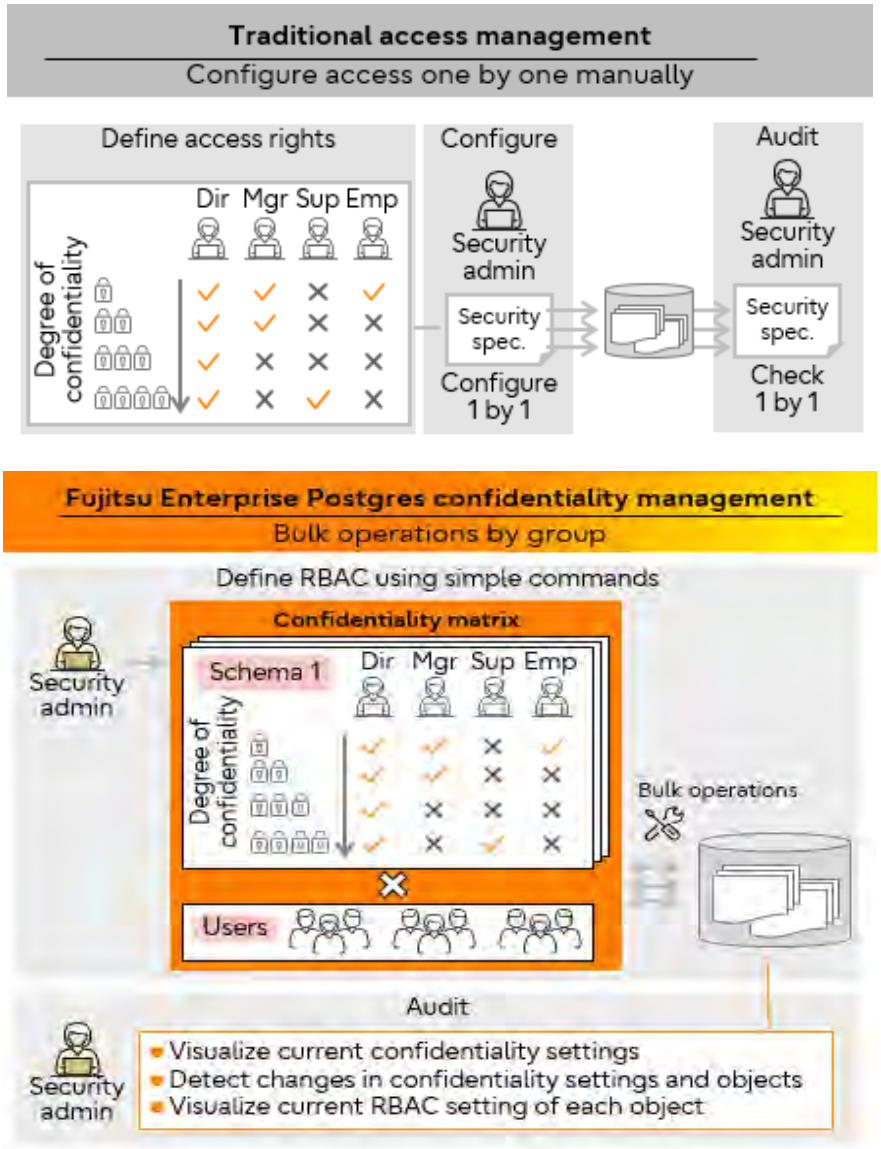
- Object-level privileges: Privileges can be assigned at various levels, including databases, schemas, tables, columns, functions, and sequences.
- Inheritance: Roles can inherit privileges from other roles, enabling hierarchical privilege management.
- Default privileges: Administrators can define default privileges for objects created within a schema, streamlining privilege management.
- Grant option: A user granted privileges with the `WITH GRANT OPTION` can further delegate those privileges to other roles.

Caveats,

- Public role: By default, all users have access to certain privileges through the PUBLIC role, which could inadvertently expose sensitive data if not configured properly.
- Privilege escalation: Mismanagement of the `WITH GRANT OPTION` can result in unauthorized users gaining higher levels of access than intended.
- Schema and search path: Unintended access can occur due to confusion between schema-level permissions and the effective search path.
- Revocation challenges: Revoking privileges does not cascade to objects derived from the revoked privileges unless explicitly specified, which might leave residual permissions.

2.3.2. Role :Based Access Control _RBAC–

The RBAC security model governs resource access based on the responsibilities assigned to people or groups. The permissions granted in RBAC are connected with certain roles rather than individual users to simplify administration and maintain uniform access control across the organization. The appropriate privileges are granted to the roles in accordance with the organization's regulations, and those roles will be assigned to the users. This may increase security while simultaneously enforcing the division of roles using this technique.



In Fujitsu Enterprise Postgres 15, the new feature introduced called confidentiality management, which provides the more streamlined and efficient approach to managing access privileges, enhancing data security and minimizing the administrative burden for the DBAs.

For more information, refer to the *Security Operation Guide × Chapter 7# Confidentiality management*.

2.3.3. Row :level security

Fujitsu Enterprise Postgres incorporates Row Level Security (RLS), a powerful feature that allows fine-grained access control over individual rows in a table based on the roles and attributes of the user executing a query. This feature is particularly useful for enforcing data privacy and regulatory compliance by ensuring that users can only access the data they are authorized to see.

With RLS, you can define security policies that restrict access to specific rows within a table. These policies are applied automatically whenever a query is executed, ensuring consistent enforcement of access controls. For example, you can create policies that allow employees to view only the records related to their department or restrict access to sensitive customer information based on user roles.

Key aspects of RLS:

- Policy definition: Administrators can define multiple security policies for a table, specifying conditions under which rows are accessible.
- Policy enforcement: Policies are enforced at the database level, ensuring that all queries, including those from applications and direct SQL access, comply with the defined rules.
- Dynamic policies: RLS policies can be dynamic, using session variables or user attributes to determine access rights at runtime.
- Integration with roles: RLS integrates seamlessly with PostgreSQL's role-based access control, allowing policies to be tailored to specific user roles and groups.

To implement RLS, you use the `CREATE POLICY` and `ALTER TABLE ... ENABLE ROW LEVEL SECURITY` commands. Once enabled, the policies are automatically applied to all queries on the table, providing robust and transparent access control.

By leveraging RLS, Fujitsu Enterprise Postgres helps organizations enhance data security, comply with regulatory requirements, and protect sensitive information from unauthorized access.

2.3.4. Data Masking

Fujitsu Enterprise Postgres Data Masking is a robust feature designed to protect sensitive data within enterprise environments. This feature is part of Fujitsu's enhanced PostgreSQL database management system, aimed at addressing the growing need for data security and compliance with regulations like GDPR and PCI-DSS.

Data masking works by obfuscating sensitive information, making it unreadable to unauthorized users while maintaining its usability for testing and development purposes. This ensures that real data is not exposed in non-production environments, significantly reducing the risk of data breaches.

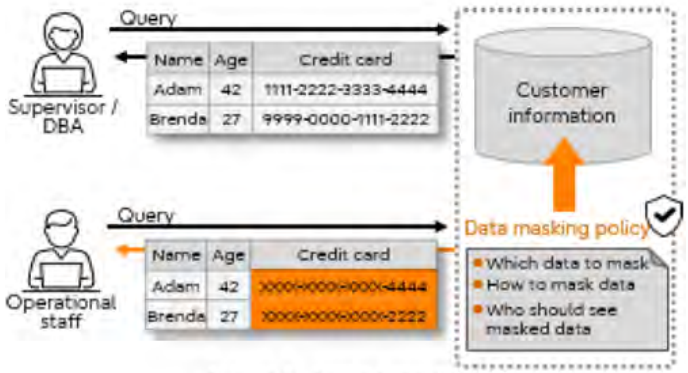


Figure 1 - Online masking

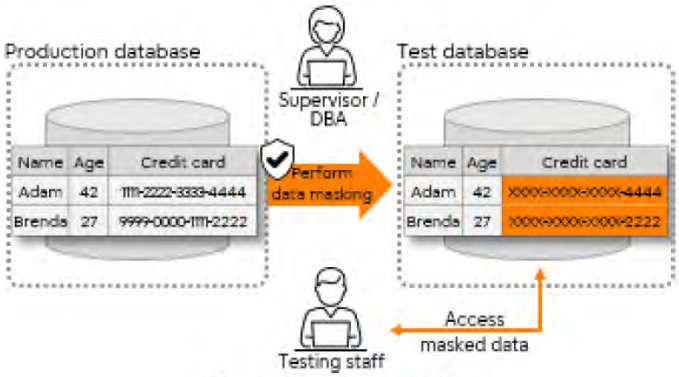


Figure 2 - Offline masking

Fujitsu's implementation of data masking allows organizations to create realistic but fictional versions of their data, which can be used safely in various scenarios without compromising the original data's integrity. This is particularly useful for companies that need to share data with third parties or conduct extensive testing without risking sensitive information.

Fujitsu Enterprise Postgres offers the three different types of data masking are available,

- Full masking: A whole column value can be obfuscated with alternate values.

```
postgres=# SELECT FROM employee;
 id | name | age | salary |     credit_card
----+------+-----+--------+---------------------
  1 | EMP1 |  42 |      0 |  1234-2345-3456-4567
  2 | EMP2 |  36 |      0 |  8686-7575-6464-5353
  3 | EMP3 |  42 |      0 |  3453-3454-5343-3433
  4 | EPP4 |  50 |      0 |  2581-4703-6925-8147
(4 rows)
```

- Partial masking: Part of a column value can be obfuscated with alternate values.

```
postgres=# SELECT FROM employee;
 id | name | age | salary |     credit_card
----+------+-----+--------+---------------------
  1 | EMP1 |  42 |      0 |  ****-****-****-4567
  2 | EMP2 |  36 |      0 |  ****-****-****-5353
  3 | EMP3 |  42 |      0 |  ****-****-****-3433
  4 | EPP4 |  50 |      0 |  ****-****-****-8147
(4 rows)
```

- Regular expression masking: The value of a column can be obfuscated via a regular expression statement.

```
postgres=# SELECT FROM employee;
 id | name | age | salary |     credit_card
----+------+-----+--------+---------------------
  1 | *1   |  42 |      0 |  ****-****-****-4567
  2 | *2   |  36 |      0 |  ****-****-****-5353
  3 | *3   |  42 |      0 |  ****-****-****-3433
  4 | *4   |  50 |      0 |  ****-****-****-8147
(4 rows)
```

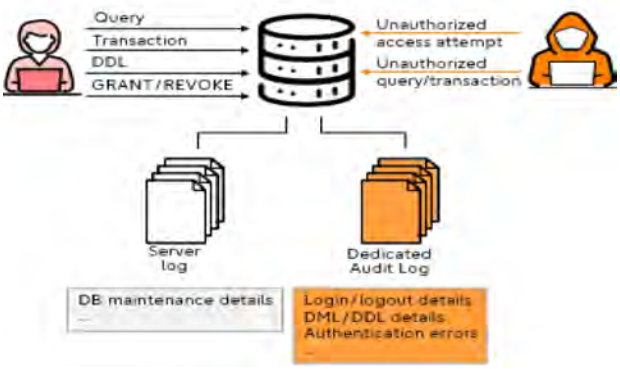
By integrating data masking, Fujitsu helps organizations mitigate the risks associated with data proliferation and unauthorized access, ensuring that sensitive data always remains protected.

2.4. Auditing

Monitoring and logging database activities is crucial for detecting and responding to security incidents. Fujitsu Enterprise Postgres provides comprehensive auditing capabilities, allowing administrators to track user actions, data changes, and access patterns. Regularly reviewing audit logs helps identify suspicious activities and enforce accountability.

2.4.1. pgAudit

The PostgreSQL Audit Extension, commonly known as pgaudit, is a powerful tool designed to enhance the auditing capabilities of Fujitsu Enterprise Postgres databases. It provides detailed sessions and object audit logging, which is essential for monitoring database activities and ensuring compliance with various regulatory standards such as government, financial, and ISO certifications.



pgaudit captures a wide range of database events, including SQL statements, user activities, and changes to database objects. This comprehensive logging helps organizations maintain a clear audit trail, making it easier to track who performed what actions, when, and how. This level of detail is crucial for detecting unauthorized access, investigating suspicious activities, and maintaining overall data integrity.

One of the standout features of pgaudit is its ability to log both successful and unsuccessful attempts to access or modify data. This dual logging capability helps identify potential security threats and ensures that all access attempts are accounted for. Additionally, pgaudit supports customizable logging, allowing administrators to specify which types of activities should be logged based on their specific security requirements.

This extension integrates seamlessly with Fujitsu Enterprise Postgres 's standard logging facility, providing robust auditing capabilities without significantly impacting database performance. It is designed to be easy to configure and use, enabling database administrators to implement comprehensive auditing practices with minimal effort. By leveraging the detailed logging capabilities of pgaudit, organizations can ensure that their database activities are transparent, traceable, and compliant with industry regulations.

To learn more, refer *PostgreSQL Insider > PostgreSQL audit log > Detecting database security threats and Heres how you get dedicated audit logging with Fujitsu Enterprise Postgres*.

2.4.2. Regular log monitoring

Regular log monitoring is a crucial practice to ensure system health and detect potential issues. By monitoring and analyzing the database logs, the DBA can identify performance bottlenecks, security breaches, and other anomalies. Additionally, implementing alert mechanisms helps promptly notify DBAs about critical events.

Effective log monitoring in Fujitsu Enterprise Postgres requires proper configuration of logging parameters in the postgresql.conf file. These parameters enable the database to record essential events, queries, and error details, providing valuable insights for performance tuning, troubleshooting, and auditing. Below are the key parameters to set up:

- **logging_collector**: Enables the logging collector process, which captures and redirects log messages to log files. This must be set to on for most logging configurations.

```
logging_collector = on
```

- **log_filename**: Sets the naming pattern for log files, typically including timestamps for easier management.

```
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log'
```

- **log_statement**: Configures which SQL statements are logged. Possible values include none, ddl, mod, and all. Setting it to all provides comprehensive logging but may impact performance.

```
log_statement = 'mod'
```

- **log_min_duration_statement**: Logs SQL statements that exceed a specified execution time. This is useful for identifying slow queries.

```
log_min_duration_statement = 1000  # Logs queries taking more than 1 second
```

- **log_checkpoints**: Logs checkpoint activity, which is crucial for understanding database performance during high loads.

```
log_checkpoints = on
```

- **log_connections** and **log_disconnections**: Logs when clients connect to or disconnect from the database, helping track usage patterns and detect anomalies.

```
log_connections = on
log_disconnections = on
```

- **log_line_prefix**: Adds contextual information like timestamps, user, database name, and session ID to each log entry, making logs more informative.

```
log_line_prefix = '%t [%p]: [%l-1] user=%u, db=%d, app=%a, client=%h '
```

- **log_temp_files**: Logs details of temporary file usage. This helps identify queries that generate large temporary files due to operations like sorting or hashing. Setting it to 0 logs all temporary files.

```
log_temp_files = 0  # Logs all temporary files
```

- **log_autovacuum_min_duration**: Logs autovacuum activities that exceed the specified duration. This aids in diagnosing performance issues caused by autovacuum processes.

```
log_autovacuum_min_duration = 1000  # Logs autovacuum tasks taking > 1 sec
```

- **log_lock_waits**: Logs queries that wait longer than the threshold defined by deadlock_timeout to acquire a lock. This is useful for identifying and resolving lock contention.

```
log_lock_waits = on
```

By configuring these parameters, you can ensure effective log monitoring in Fujitsu Enterprise Postgres, enabling timely detection of issues, better performance analysis, and improved operational insights.

3. Enable encryption for data protection

Data encryption is a critical process in securing sensitive information by transforming readable data, known as plaintext, into an unreadable, scrambled format called ciphertext. This transformation is performed using advanced encryption algorithms and encryption keys, which work together to ensure that the data remains unintelligible to unauthorized individuals.

Encryption serves as a robust defence mechanism against unauthorized access, eavesdropping, and data breaches. Even if encrypted data is intercepted during transmission or stolen from storage, it remains protected because the ciphertext cannot be deciphered without the correct decryption key. This ensures that only authorized parties, who possess the appropriate key, can reverse the encryption process and access the original plaintext.

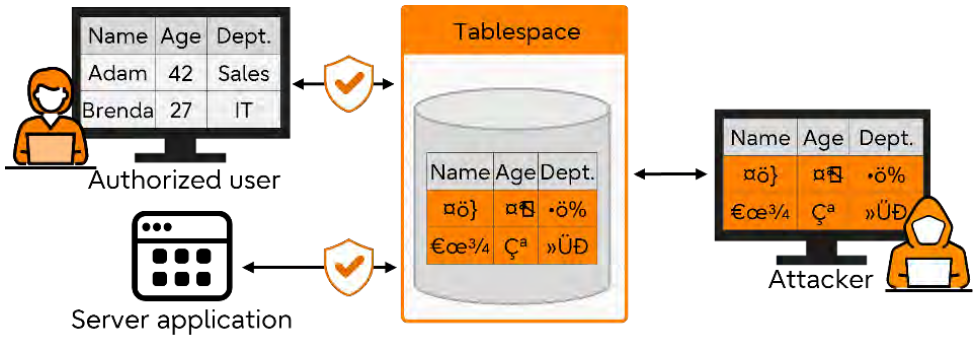
3.1. Transparent Data Encryption _TDE–

Fujitsu Enterprise Postgres Transparent Data Encryption (TDE) is a robust security feature designed to protect data at rest by encrypting database files and logs. This ensures that sensitive information remains secure even if the storage media is compromised. TDE uses 256-bit Advanced Encryption Standard (AES) encryption, which is compliant with stringent data protection regulations like GDPR and PCI-DSS.

One of the key advantages of TDE is that it encrypts data automatically as it is written to the disk and decrypts it when read, without requiring any changes to existing applications. This seamless integration means that organizations can enhance their data security without disrupting their operations or incurring significant additional costs.

Fujitsu Enterprise Postgres TDE also supports the encryption of individual tablespaces and logs, each with their own encryption keys. This granular level of encryption ensures that even if one part of the database is compromised, the rest remains secure. Additionally, TDE minimizes encryption and decryption overhead, ensuring that database performance is not significantly impacted.

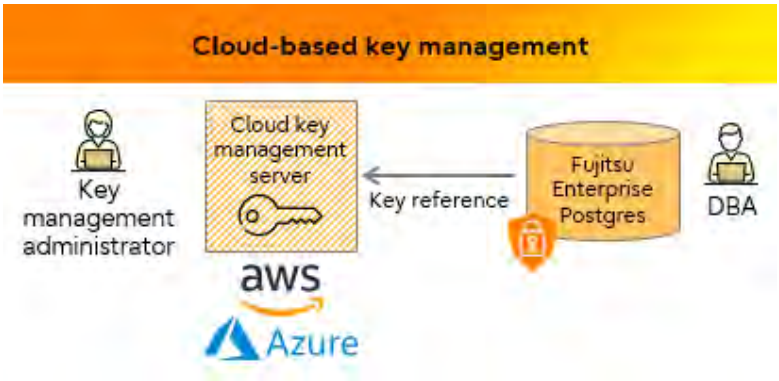
By implementing TDE, organizations can protect their data from unauthorized access, reducing the risk of data breaches, and ensure compliance with various regulatory requirements. This makes Fujitsu Enterprise Postgres TDE an essential tool for any enterprise looking to enhance its data security posture.



It is also important to understand that when disks fail or are taken out of service, they are often discarded, sent for recycling, or end up in a landfill. Even in these cases, there is still a high chance that someone could recover and misuse the data stored on them if it is not properly encrypted or destroyed. This is a serious risk because old disks can contain sensitive business or customer information. By using TDE, organizations can make sure that all data on these disks is encrypted. This means that even if someone gets hold of the physical disk, they will not be able to read or use the data without the proper encryption keys. This extra layer of security helps protect valuable information from being exposed, even after the disk is no longer in use.

To know more about the TDE in Fujitsu Enterprise Postgres, refer *Securing your data with Transparent Data Encryption* and *Providing maximum data security with minimal impact to your business using transparent data encryption*.

In Fujitsu Enterprise Postgres 15, the new feature introduced ***cloud :based key management for TDE*** which provides the robust cloud-based key management services with TDE enhancing data security and compliance. This system leverages cloud key management services (KMS) to store encryption keys outside the database, significantly reducing the risk of data leakage. By separating the roles of DBAs and key administrators, it ensures better governance and control over sensitive data.



The key management system is KMIP-certified, which means it adheres to industry standards for managing encryption keys. This certification ensures interoperability with various KMS providers, offering flexibility and scalability for enterprises. Additionally, the system supports automated key rotation and lifecycle management, which helps to main the security of encrypted data over time.

For more information, refer to the *Operation Guide > Chapter 6 : Using Transparent Data Encryption with Key Management Systems as Keystores*.

3.2. pgcrypto

Fujitsu Enterprise Postgres pgcrypto is a powerful cryptographic extension designed to enhance data security within PostgreSQL databases. This module provides a range of cryptographic functions, including hashing, encryption, and decryption, which are essential for protecting sensitive data.

The pgcrypto module supports various cryptographic algorithms such as MD5, SHA (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512), HMAC, AES, and BLOWFISH. These algorithms enable users to securely hash passwords, encrypt data, and ensure data integrity. For instance, hashing passwords with pgcrypto involves adding a random salt to each password before hashing, making it significantly harder for attackers to crack the passwords.

One of the key benefits of pgcrypto is its ability to perform both one-way and two-way encryption. One-way encryption, or hashing, is useful for storing passwords securely, as it converts the password into a fixed-length string that cannot be easily reversed. Two-way encryption, on the other hand, allows data to be encrypted and later decrypted, which is useful for protecting sensitive information that needs to be accessed in its original form.

Fujitsu Enterprise Postgres ensures that pgcrypto is integrated seamlessly, providing robust security features without compromising performance. The module is designed to be easy to use, allowing database administrators and developers to implement strong encryption practices with minimal effort.

For more information, refer *How to use pgcrypto to further protect your data*.

4. Conduct regular vulnerability assessments and penetration testing

Regular vulnerability assessments and penetration testing are vital for ensuring security. These assessments help identify and address potential security flaws promptly in your Fujitsu Enterprise Postgres database.

4.1. Vulnerability scanning

Comprehensive scans and tests are used in these assessments to identify vulnerabilities, misconfigurations, and obsolete software components. Organizations can fix vulnerabilities and reduce the risk of cyberattacks by doing this on a regular basis. This also assures that data saved in Fujitsu Enterprise Postgres remains confidential, intact, and available.

As a result, the assessment report leads the adoption of appropriate patches, upgrades, and security measures to improve overall database system security in accordance with industry standards.

4.2. Penetration testing

This is a vital security activity that extends beyond vulnerability assessments. Controlled penetration testing sessions will be carried out by proficient security experts. These tests simulate real-world attack situations in order to find flaws that may go undetected by automated scans alone. This test assists the company in assessing the efficiency of its security procedures and detecting any loopholes that malicious attackers may exploit. It also assists the company in making the right decision about risk mitigation and necessary remediation measures.

5. Continuous security awareness and training

Continuous security awareness and training are essential for hardening an organization's infrastructure and systems. It involves implementing a proactive strategy to continuously educate employees and other stakeholders on security risks, best practices, and potential threats. Organizations may build a security-conscious culture and empower employees to make educated decisions that safeguard sensitive information and assets by implementing a continuous security awareness program.

The following are the important components of continual security awareness and training:

5.1. Education and training

Regular training sessions and workshops are conducted to educate staff about different security concerns such as phishing, social engineering, malware, and password security. Data classification, secure coding methods, incident response, and regulatory compliance may also be included in training sessions. Online courses, workshops, newsletters, and interactive simulations can all be used to provide these sessions.

5.2. Threat awareness

Continuous security awareness focuses on keeping employees informed about the most recent security threats and methods of attack. To distribute information about potential risks, previous security events, and lessons learned, regular communication channels such as email newsletters, security bulletins, and internal security blogs are used. This enables users to more fully understand the evolving threat landscape and to detect and report possible security problems.

5.3. Security policies and procedures

Enforcing security policies and procedures is a critical component of maintaining continuous security awareness. Organizations should review and update their security policies on a regular basis to ensure that they are in line with new threats and industry best practices. Employees should be aware of the guidelines and understand their roles in ensuring security. The need to comply with regulations regarding data protection, access controls, incident reporting, and the appropriate use of technical resources should be emphasized throughout training.

5.4. Incident response and reporting

Employees should be trained on how to report security incidents and possible vulnerabilities. To ensure that employees are prepared to respond successfully in the case of a security breach, regular training should include incident response simulations and exercises. Understanding how to identify and report suspicious activity, how to contain a breach, and how to interact with key security departments inside the company are all part of this.

5.5. Ongoing evaluation and metrics

Methods for monitoring the effectiveness of continuous security awareness programs should be included. Periodic evaluations and surveys can be conducted by organizations to assess workers' understanding of security practices and highlight areas for improvement. Metrics such as phishing simulation click rates, incident reaction times, and the number of reported security issues provide insight into overall security posture and training success.

6. Implementation of CIS benchmark

The CIS benchmark is developed through a collaborative process by involving the various experts for the fields. These experts come from various fields like consulting, software development, audit & compliance, security research, legal sectors and many more. These kinds of diverse perspectives contribute to well comprehensive benchmark.

This benchmark for PostgreSQL assesses the performance and security of PostgreSQL database systems. It establishes a set of guidelines and best practices to evaluate the database configuration, access control and overall other functionalities like replication, backup, and etc. to meet the industry standards and compliance policies. By adhering to these benchmarks organizations can strengthen their security and reduce the risks of common vulnerabilities.

Regular benchmarking against CIS guidelines can help to maintain a robust and well-protected database infrastructure.

The CIS benchmarking guidelines for PostgreSQL can be found at *CIS PostgreSQL Benchmarks*

Conclusion

To protect your organization's critical information, it is essential to secure your Fujitsu Enterprise Postgres database. Implementing the comprehensive security measures outlined in this document will significantly enhance the security posture of your Fujitsu Enterprise Postgres environment. Additionally, it is crucial to consider encrypting your database backups, especially if they are stored offsite or transmitted externally, to safeguard data confidentiality beyond the primary system. To ensure ongoing protection, adopt a proactive approach, stay adaptable to emerging threats, and continuously assess and update your security protocols.

FUJITSU

Fujitsu Enterprise Postgres is the enhanced version of PostgreSQL, for enterprises seeking a more robust, secure, and fully supported edition for business-critical applications

Contact

Fujitsu Limited

Email: enterprisepostgresql@fujitsu.com

Website: fast.fujitsu.com