Member-only story

Enhance PostgreSQL Security: A Guide to the Passwordcheck Extension

Sheikh Wasiu Al Hasib · Following

3 min read · Nov 21, 2024

Listen Share More

`passwordcheck` is a PostgreSQL extension designed to enforce password complexity policies when setting passwords for database users. It helps ensure that passwords meet specific security standards, making it harder for attackers to guess or compromise them.

Key Features of `passwordcheck`

1. Password Complexity Enforcement:
— Enforces rules like minimum length, use of special characters, and prohibiting passwords similar to usernames.

2. Customization:
— Allows database administrators to define and enforce custom password policies.

3. Integration with PostgreSQL:
— Operates at the database level and is triggered automatically during password changes (e.g., via `CREATE ROLE` or `ALTER ROLE` commands).

4. Prevention of Weak Passwords:
— Prevents setting weak passwords that could compromise database security.

How `passwordcheck` Works

- When the `**passwordcheck**` extension is installed and activated, it intercepts password-setting operations.

- If the password does not meet the defined complexity criteria, the operation fails with an error message.
- The criteria are configured in the PostgreSQL configuration file (`**postgresql.conf**`).

Steps to Use `passwordcheck`

1. **Install the Extension**

   `**passwordcheck**` is part of the PostgreSQL contrib package, so ensure it is installed on your system:

```
sudo yum install postgresql-contrib # For RHEL-based systems
sudo apt install postgresql-contrib # For Debian-based systems
```


```
shared_preload_libraries = 'passwordcheck'
```

2. Restart PostgreSQL to apply the changes:

```
sudo systemctl restart postgresql
```

3. Test the Extension

Set a new password for a role to ensure the password policy is enforced:

```
ALTER ROLE username PASSWORD 'weakpass';
 — Output: ERROR: password is too short
```

Configuring `passwordcheck`

You can configure `**passwordcheck**` in the PostgreSQL configuration file (`**postgresql.conf**`) to define specific password rules. Here are the commonly enforced policies:

1. **Minimum Password Length:**
   Enforce a minimum number of characters for passwords.

```
passwordcheck.min_length = 8
```

2. Deny Password Similar to Username:
Prevent passwords that are identical to or derived from the username.

```
passwordcheck.deny_similar_to_username = on
```

3. Special Characters Requirement:
Enforce the presence of special characters, numbers, or uppercase letters.
This may require customization (e.g., by modifying the extension code if the built-in policies don't suffice).

Using `passwordcheck` for Security

1. Enforce Strong Passwords:
Ensure that all database users set passwords that meet security requirements.

2. Compliance with Security Policies:
Use `**passwordcheck**` to align with organizational or regulatory standards for password management (e.g., PCI DSS, HIPAA).

3. Integrate with Auditing:
Combine `**passwordcheck**` with PostgreSQL auditing tools to monitor password changes and enforce security practices.

Limitations

1. Limited Customization:
— Advanced password policies (e.g., requiring specific character classes) may not be fully supported without modifying the extension code.

2. Does Not Encrypt Passwords:
— PostgreSQL hashes passwords using its built-in mechanisms. Use `passwordcheck` alongside secure transport mechanisms (e.g., SSL/TLS) to secure password transmissions.

3. Applicability:
— Only affects passwords managed directly in PostgreSQL. If you're using external authentication (e.g., LDAP, Kerberos), `passwordcheck` will not apply.

Example Usage

Enforcing a Secure Password Policy
1. Enable `passwordcheck`:

```
shared_preload_libraries = 'passwordcheck'
passwordcheck.min_length = 12
passwordcheck.deny_similar_to_username = on
```

2. Restart PostgreSQL:

```
sudo systemctl restart postgresql
```

3. Attempt to set a password:

```
ALTER ROLE user PASSWORD 'simplepass';
 — Output: ERROR: password is too short
```

```
ALTER ROLE user PASSWORD 'Complex$Pass123';
 — Password updated successfully
```

Conclusion

`**passwordcheck**` is a straightforward yet powerful tool for improving the security of PostgreSQL databases by enforcing strong password policies. By integrating it with PostgreSQL's authentication mechanisms, it helps organizations comply with security best practices and reduce the risk of unauthorized access.

Passwordcheck | Postgersql Password Auth | Postgresql Security

Following

Written by Sheikh Wasiu Al Hasib

80 Followers · 3 Following

I am a PostgreSQL DBA with RHCE,RHCSA, Hardening ,CKA and CKS certified. I enjoy to work with database, automation tools like ansible ,terraform, bash etc. .

No responses yet

Gvadakte

What are your thoughts?

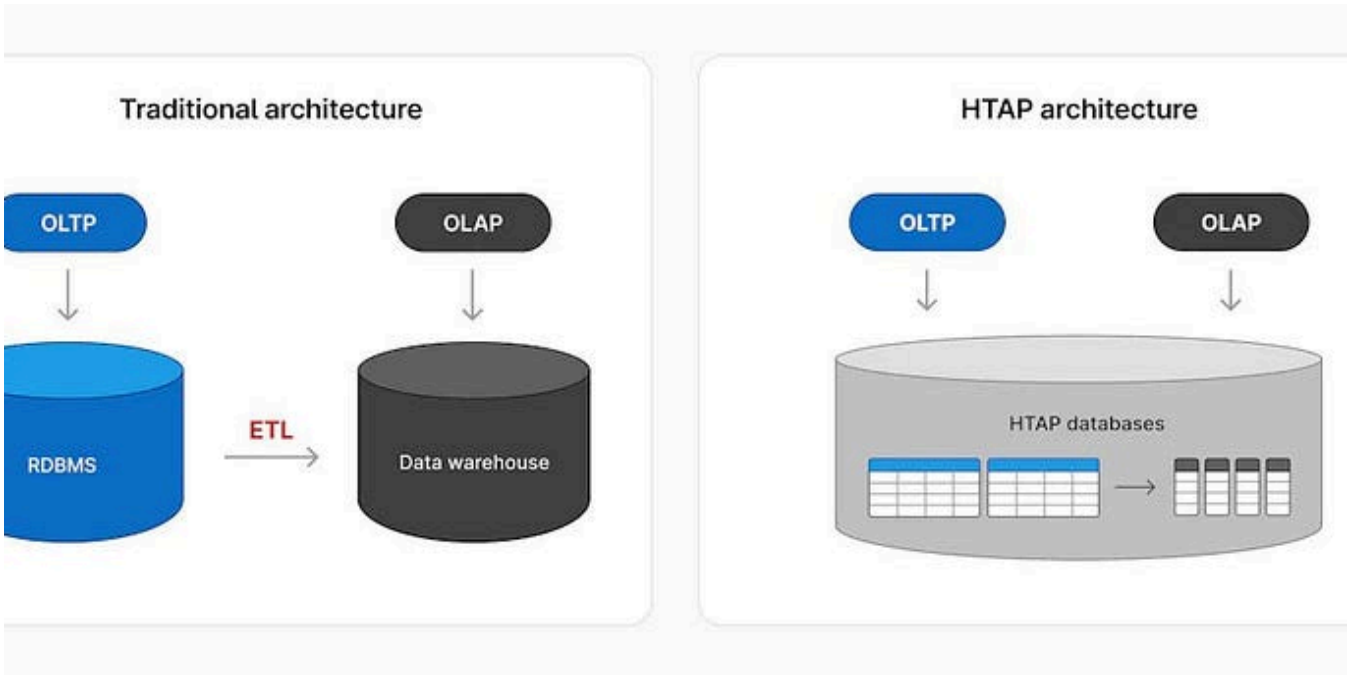
More from Sheikh Wasiu Al Hasib

Sheikh Wasiu Al Hasib

Effect of multiple update on same row in PostgreSQL tables

In PostgreSQL, updates to the same row in a table do not lock the whole table; instead, they lock the specific row being updated. Here's...

Aug 2, 2024 6



Sheikh Wasiu Al Hasib

PostgreSQL Hybrid Transactional/Analytical Processing using

What is HTAP?

Jul 16, 2024 6

Sheikh Wasiu Al Hasib

Balancing Disk Space and Reliability with PostgreSQL's max_slot_wal_keep_size

The `max_slot_wal_keep_size` parameter in PostgreSQL is a configuration setting that helps manage the retention of WAL (Write-Ahead...

Aug 4, 2024 1

Sheikh Wasiu Al Hasib

Different method of replicating data from PostgreSQL to Clickhouse as a Reporting Database

Choosing between ClickHouse and PostgreSQL depends largely on the specific use cases and the nature of the workloads they need to support...

Aug 26, 2024 1

See all from Sheikh Wasiu Al Hasib

Recommended from Medium

In Dev Genius by Doran Gao

Efficient Large Table Cleanup in PostgreSQL

“We can only see a short distance ahead, but we can see plenty there that needs to be done.” —Alan Turing

Oct 31, 2024 1

Ajaymaurya

How Often Should You Reindex Your PostgreSQL Database? A Data-Driven Approach

PostgreSQL is one of the most powerful and flexible relational databases available today. However, like any database system, it requires...

Mar 24

In Databases by Sergey Egorenkov

Making SQL query 40x faster for 10 million rows table

Make your SQL query really fast using this approach

Mar 17 10

In Towards Dev by Nakul Mitra

PostgreSQL Performance Optimization—Cleaning Dead Tuples & Reindexing

Performance optimization is crucial in PostgreSQL to ensure efficient query execution and minimal resource consumption.

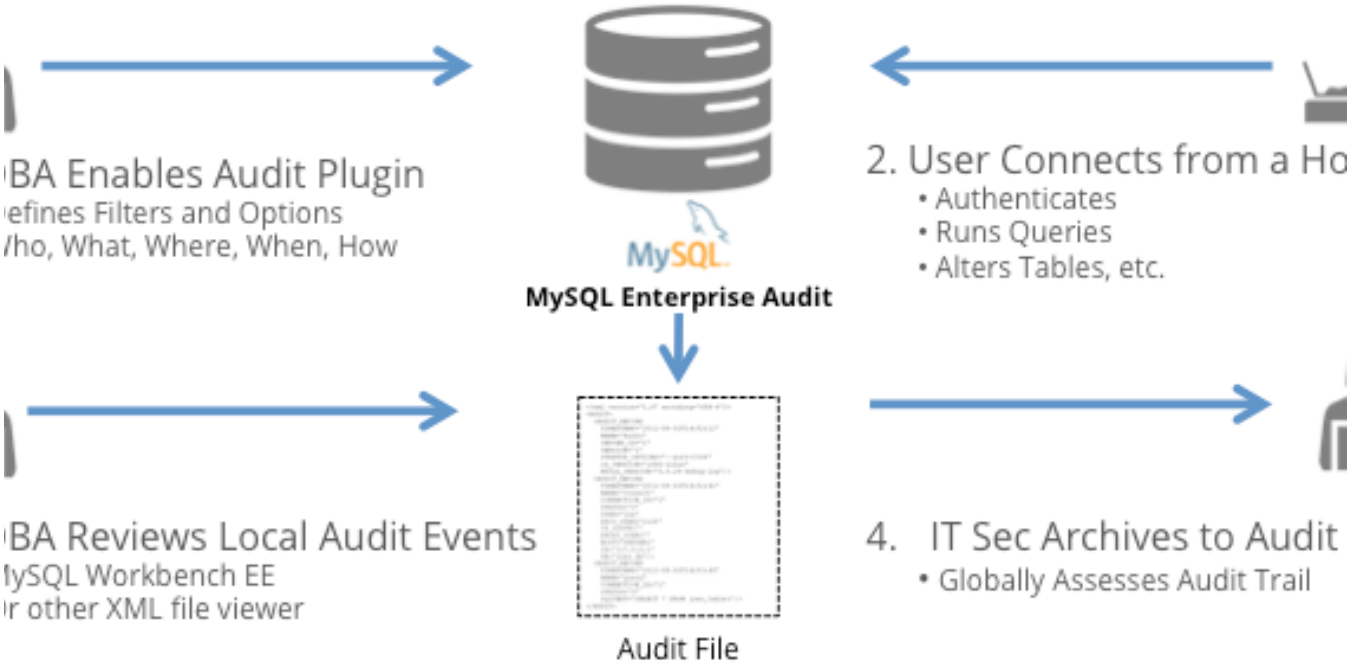
Mar 28 1

In Hack the Stack by Coders Stop

9 Database Optimization Tricks SQL Experts Are Hiding From You

Most developers learn enough SQL to get by—SELECT, INSERT, UPDATE, DELETE, and maybe a few JOINs. They might even know how to create...

Mar 27 208 5



Hari

How to Enable MYSQL DB audit logs

Step 1: Install the MySQL Enterprise Audit Plugin

Nov 12, 2024 53

See more recommendations