Pontificia Universidad Javeriana

Fundamentos de ingeniería de software

Greenet

Documento investigativo

Encriptación de contraseñas con Bytea en PostgreSQL

A cargo de Mateo Zamora Pérez

Pontificia Universidad JAVERIANA Bogotá

Contenido

Introducción

La seguridad en el manejo de contraseñas es un aspecto muy importante en el diseño de sistemas que almacenan información. PostgreSQL tiene mecanismos para almacenar y procesar contraseñas de manera segura, mediante el módulo pgcrypto, que permite aplicar funciones de hashing y cifrado sobre columnas de tipo bytea.

Este documento analiza las prácticas recomendadas para almacenar contraseñas en PostgreSQL utilizando bytea.

Hashing vs cifrado

Es necesario diferenciar el hashing y el cifrado para comprender la seguridad de contraseñas. El cifrado (encryption) es un proceso reversible en el que la información se transforma en un formato codificado, que puede recuperarse con una clave. Es útil para datos que deben ser recuperados, pero no es recomendable para contraseñas (Anderson, 2020).

El hashing es un proceso unidireccional que transforma las contraseñas en cadenas de longitud fija. En este proceso, el sistema no guarda la contraseña original, sino su representación en hash. Para la verificación, se aplica el mismo algoritmo a la contraseña ingresada y se compara el resultado. Para las contraseñas se usan algoritmos adaptativos como bcrypt, scrypt o Argon2, diseñados para resistir ataques de fuerza bruta y rainbow tables[1] (Bonneau & Preibusch, 2010).

PostgreSQL y el módulo

PostgreSQL incluye el módulo pgcrypto, que proporciona funciones de hashing y cifrado. Para contraseñas, las funciones más utilizadas son:

- crypt() y gen_salt(): diseñadas para generar hashes seguros, salados[2] y resistentes a ataques de fuerza bruta.

[1] Proceso de hackeo que utiliza listas de contraseñas comunes y calcula sus hashes (Schrader, 2025)
[2] Salting añade un dato aleatorio a la contraseña antes del hash (NordPass Blog)

- digest(): permite aplicar funciones hash como SHA-256 o SHA-512 y almacenar el resultado en una columna de tipo bytea.

Bytea

Bytea es un tipo de dato que es muy útil cuando se manejan datos binarios obtenidos por operaciones criptográficas. Al almacenar el hash en bytea, se evita la pérdida de información en conversiones de texto y se conserva el resultado en su forma original.

Este es un ejemplo del uso de digest() con SHA-256:

```
CREATE TABLE usuarios (
  id SERIAL PRIMARY KEY,
  usuario TEXT NOT NULL,
  password_hash BYTEA NOT NULL
);

INSERT INTO usuarios (usuario, password_hash)
VALUES ('ana', digest('mi_contraseña', 'sha256'));
```

Observaciones útiles aplicables a Greenet

Es importante que usemos hashing como protección de contraseñas en PostgreSQL y no mediante cifrado. También, deberíamos usar el tipo de dato bytea para almacenar resultados binarios criptográficos.

Referencias

1. Anderson, R. (2020). Security Engineering: A Guide to Building Dependable Distributed Systems. https://www.researchgate.net/publication/234775191_Security_Engineering_A_Guide_to_Building_Dependable_Distributed_Systems
2. Bonneau, J., & Preibusch, S. (2010). The password thicket: technical and market failures in human authentication on the web. https://www.researchgate.net/publication/242747511_The_Password_Thicket_technical_and_market_failures_in_human_authentication_on_the_web
3. NordPass. ¿Qué es una sal de contraseña? https://nordpass.com/es/blog/password-salt/
4. PostgreSQL Global Development Group. (2023). pgcrypto Documentation. Recuperado de https://www.postgresql.org/docs/current/pgcrypto.html
5. PostgreSQL Pro. (2023). pgcrypto Module Overview. Recuperado de https://postgrespro.com/docs/postgresql/current/pgcrypto
6. RCS Internet Services. (2022). How to Securely Store Passwords Using PostgreSQL. Recuperado de https://rcs.is/knowledgebase/36
7. Schrader, D. (2025, 21 junio). Rainbow Table Attacks: How They Work and How to Defend Against Them. netwrix. https://blog.netwrix.com/rainbow-table-attack