Aula: Criptografia de Senhas no Banco de Dados MySQL

Entender a importância da criptografia de senhas para a segurança dos dados.

Conhecer os métodos de criptografia mais comuns utilizados para senhas.

Aprender a aplicar a criptografia de senhas no MySQL.

Parte 1: Introdução à Criptografia de Senhas

O Que é Criptografia?

A criptografia é uma prática milenar, evoluída ao longo dos séculos, que se refere à técnica de codificar mensagens ou informações de maneira que apenas as partes autorizadas possam acessá-las. Em sua essência, a criptografia transforma dados originais, conhecidos como texto claro, em uma forma codificada, conhecida como texto cifrado, através de algoritmos complexos e chaves de criptografia. Esse processo visa garantir a confidencialidade, integridade e autenticidade das informações, protegendo-as contra acessos não autorizados ou modificações indesejadas.

Por Que a Criptografia é Crucial para a Segurança das Senhas?

As senhas são a linha de frente na proteção de nossas informações pessoais e acessos a sistemas diversos. A criptografia desempenha um papel crucial na segurança das senhas, pois:

Proteção contra Acessos Não Autorizados: Ao criptografar senhas, mesmo que um invasor consiga acessar o banco de dados onde elas estão armazenadas, ele encontrará apenas texto cifrado, ininteligível sem a chave de descriptografia correta.

Integridade dos Dados: A criptografia garante que as senhas armazenadas não sejam alteradas ou manipuladas, assegurando que a autenticação do usuário seja sempre realizada com os dados originais.

Confidencialidade: Assegura que as informações sensíveis, como senhas, permaneçam privadas e acessíveis apenas por aqueles que possuem as chaves corretas, seja numa troca de informações entre sistemas ou no armazenamento desses dados.

Em resumo, a criptografia é uma ferramenta indispensável na proteção das senhas, agindo como um escudo contra tentativas de violação de dados e garantindo a segurança das informações em um mundo digital cada vez mais vulnerável a ataques cibernéticos.

Parte 2: Métodos de Criptografia de Senhas

HASH

O que é função hash e como ela funciona?

Uma função hash é um algoritmo que recebe uma entrada (ou "mensagem") de qualquer tamanho e produz uma saída de tamanho fixo, geralmente referida como "hash" ou "digest". As funções hash são projetadas para serem unidirecionais, o que significa que é computacionalmente inviável reverter o processo e recuperar a entrada original a partir do hash. Além disso, uma pequena alteração na entrada (mesmo uma mudança de um único caractere) produzirá um hash completamente diferente, um conceito conhecido como "efeito avalanche".

As funções hash são utilizadas em uma variedade de aplicações de segurança da informação, como a verificação de integridade de dados, autenticação e armazenamento seguro de senhas. Quando usadas para senhas, as funções hash permitem que sistemas armazenem e verifiquem credenciais sem a necessidade de manter as senhas reais armazenadas de forma legível.

Diferença entre hash e criptografia

Embora tanto o hash quanto a criptografia sejam usados para transformar dados, eles servem a propósitos distintos e operam de maneira diferente:

Criptografia é o processo de codificar informações de forma que apenas partes autorizadas possam acessá-las, usando uma chave para criptografar e descriptografar a mensagem. Seu objetivo principal é garantir a confidencialidade dos dados. A criptografia é reversível, desde que você possua a chave correta.

Hash é o processo de transformar informações em um digest de tamanho fixo que representa a entrada de forma única. As funções hash são projetadas para serem unidirecionais, tornando-se impraticável reverter o hash para a entrada original. O propósito principal do hash não é a confidencialidade, mas sim a integridade dos dados e a autenticação.

Exemplos de algoritmos de hash: MD5, SHA-1, SHA-256

MD5: O MD5 (Message Digest Algorithm 5) gera um hash de 128 bits (32 caracteres). Embora tenha sido amplamente usado, hoje é considerado inseguro para a maioria dos propósitos devido à vulnerabilidade a colisões (duas entradas diferentes produzindo o mesmo hash).

SHA-1: O Secure Hash Algorithm 1 produz um hash de 160 bits (40 caracteres). Assim como o MD5, o SHA-1 foi comprovadamente vulnerável a ataques de colisão, o que

compromete sua segurança e o torna inadequado para aplicações que exigem alta integridade dos dados.

SHA-256: Parte da família SHA-2 de algoritmos de hash, o SHA-256 gera um hash de 256 bits (64 caracteres) e é amplamente adotado por sua robustez e resistência a ataques. É recomendado para novas implementações de segurança, oferecendo um bom equilíbrio entre desempenho e segurança.

Cada um desses algoritmos tem sido utilizado em diversos contextos, mas com o passar do tempo e o avanço das capacidades computacionais, a comunidade de segurança recomenda o uso de algoritmos mais fortes e seguros, como o SHA-256, para garantir a integridade e a segurança dos dados.

Parte 3: Criptografia de Senhas no MySQL

Armazenando Senhas com HASH no MySQL

No MySQL, você pode aplicar funções hash diretamente em suas consultas e operações de banco de dados. Utilizar uma função hash como SHA-256 no MySQL é uma prática comum para aumentar a segurança ao armazenar senhas, pois mesmo que os dados do banco de dados sejam comprometidos, as senhas originais dos usuários não serão expostas.

Para aplicar a função hash SHA-256 no MySQL, você usaria a função SHA2(), que aceita dois argumentos: a string a ser hasheada e o número de bits do hash desejado (256 para SHA-256).

Exemplo de código SQL para criar e verificar senhas hash

Criando senhas hash

Quando um novo usuário se registra e escolhe uma senha, você armazena no banco de dados não a senha em si, mas o hash dessa senha. Veja como você pode fazer isso usando SHA-256:

INSERT INTO usuarios (nome, senha_hash)

VALUES ('nomeDoUsuario', SHA2('senhaDoUsuario', 256));

Verificando senhas hash

Para verificar a senha durante o login, você não tentará "descriptografar" o hash. Em vez disso, aplicará a mesma função hash à senha fornecida pelo usuário no momento do login e comparará o resultado com o hash armazenado no banco de dados. Se os hashes coincidirem, a senha está correta.

SELECT nome

FROM usuarios

WHERE nome = 'nomeDoUsuario' AND senha_hash = SHA2('senhaFornecidaNoLogin', 256);

Se a consulta retornar o registro do usuário, isso significa que a senha fornecida é correta, pois o hash gerado a partir dela corresponde ao hash armazenado no banco de dados. Caso contrário, a senha é incorreta.

Utilizando Funções de Criptografia Externas

Por que o MySQL pode não ser a melhor ferramenta para aplicar algumas criptografias, como Bcrypt.

Como usar linguagens de programação (PHP, Python, etc.) para criptografar senhas antes de armazená-las no MySQL.

Melhores Práticas

Nunca armazenar senhas em texto puro.

Utilizar algoritmos de hash modernos e seguros.

Implementar salting e considerar o uso de Bcrypt para maior segurança.

Atividade

Criar a tabela:

```
CREATE TABLE usuarios (
  id INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100),
  senha_hash CHAR(64) -- SHA-256 produz um hash de 64 caracteres
);
```

Inserir dados:

]

```
INSERT INTO usuarios (nome, senha_hash)
VALUES ('usuarioExemplo', SHA2('senha123', 256));
```