Práticas de Banco de Dados

Parte 2 – Segurança em Banco de Dados

Professor Eduardo Xavier

Segurança e Autorização

- Foco: **Proteger o banco de dados contra acessos não autorizadas**
- Envolve questões que incluem:
  - Questões legais, éticas e de direito ao acesso à informação;
  - Questões políticas a nível governamental, institucional ou corporativo;
  - Questões relacionadas ao nível do sistema (que funções de segurança devem ser implementadas?)
  - Necessidades das organizações relativas a níveis de segurança: (altamente confidencial (top secret), secreto (secret), confidencial (confidential) e não confidencial (unclassified))

Segurança e Autorização

- **Objetivos da segurança (sob constante ameaça)**
  - **Perda de integridade** – proteção contra modificação imprópria. A integridade é perdida a partir de modificações intencionais ou não;
  - **Perda de disponibilidade** – Manter o BD disponível para usuário ou programa que tem direito legítimo a ele;
  - **Perda de confidencialidade** – Proteção do BD contra divulgação não autorizada. Pode implicar em questões legais e também perda de confiança pública e constrangimentos.

Segurança e Autorização

- **O papel do DBA (Data Base Administrator)**
  - Concessão de privilégios a usuários autorizados
  - Classificação dos usuários e dados de acordo com a política da organização
  - Criação de contas
  - Revogação de privilégios
  - Atribuição de nível de segurança
  - Auditoria do banco de dados, a partir do LOG

Segurança e Autorização

- **Medidas para proteção do BD**
  - Controle de acesso
    - Mecanismos de acesso discricionário
    - Mecanismos de acesso obrigatório (mandatory)
  - Controle de inferência
    - Segurança estatística de BDs
  - Criptografia

Segurança e Autorização

- **Controle de acesso DISCRICIONÁRIO**
  - Concessão e revogação de privilégios
  - Tipos de privilégios discricionários:
    - **Nível de conta** – privilégios que a conta tem, independente das relações existentes no BD
      - Pode incluir CREATE SCHEMA, CREATE TABLE, CREATE VIEW, DROP, etc.
      - Se aplicam à conta de maneira genérica.
    - **Nível de relações** – O DBA pode controlar privilégio para acessar cada relação ou visão individual.
      - Especifica para cada usuário as relações individuais nas quais cada tipo de comando pode ser aplicado. Sempre é atribuída uma conta de proprietário. Quem possui a conta proprietário pode repassar privilégios, como Privilégio SELECT (para recuperação) ou MODIFY (para UPDATE).

Segurança e Autorização

- **Visões são mecanismos importantes de autorização discricionária.**
  - Exemplo: Uma conta “X” deve ser capaz de recuperar apenas alguns campos da relação “R”, então pode-se criar uma **visão** que inclua apenas os atributos e conceder SELECT na visão para “X”.

- Privilégios discricionários são atribuídos normalmente através das instruções de DDL:
  - **GRANT**
  - **REVOKE**

Segurança e Autorização

- **GRANT / REVOKE**
  - Cada objeto do banco de dados tem um “dono” (owner), que é seu criador.
  - Apenas o criador ou dono pode acessar os objetos
  - SQL oferece um esquema de permissões através dos comandos GRANT / REVOKE

Segurança e Autorização

GRANT

- Permissão de comandos DDL

  GRANT {comando} TO {usuário}

- Permissão de objeto

  GRANT {comando} ON {objeto} TO {usuário} [WITH GRANT OPTION]

OBS1: “WITH GRANT OPTION permite que o usuário tenha poderes de conceder GRANTS sobre estes objetos para outros usuários

OBS2: Essa não é a sintaxe completa do comando, apenas um recorte parcial que contempla o interesse da disciplina. Para mais detalhes de sintaxe, consulte o manual de SQL do banco de dados você estiver utilizando.

Segurança e Autorização

GRANT (Exemplos)

GRANT ALTER TABLE TO usuario1;

GRANT SELECT, INSERT, UPDATE, DELETE ON empregados TO usuario1;

GRANT ALL ON empregados TO usuario1;

GRANT SELECT ON empregados (nome, codprojeto) TO public;
(sintaxe do SQL Server)

GRANT SELECT (nome, codprojeto) ON empregado TO public;
(sintaxe do ORACLE)

Segurança e Autorização

REVOKE

- Retira privilégios

REVOKE {comando} ON {objeto} FROM {usuário}

REVOKE (Exemplos)

REVOKE DELETE ON empregados FROM maria;

REVOKE ALL ON empregados FROM marcelo;

REVOKE ALL ON empregados FROM public;

Segurança e Autorização

SHOW

- No MySQL, exibe privilégios de um usuário

  SHOW GRANTS FOR fulano@localhost;

OBS: No MS SQL Server os privilégios são vistos por meio de stored procedures da “família” SP_HELP.

Segurança e Autorização

- **Controle de acesso OBRIGATÓRIO (MANDATORY)**
  - Implementa uma segurança multinível.
  - Estabelece classes como ***top secret*** **(AS)**, ***secret*** **(S)**, ***confidential*** **(C)** e ***unclassified*** **(NC)**, onde **AS** é o nível mais alto.
  - Classifica cada **sujeito** (usuário, conta, programa) e **objeto** (relação, tupla, coluna, visão, operação) em uma classificação:
    - **Propriedade de segurança simples** - Um sujeito não tem permissão de acesso sobre um objeto a menos que sua classe(Sujeito) >= classe(Objeto).
    - **Propriedade estrela –** Um sujeito não tem permissão de escrever um objeto a menos que classe(Sujeito) <= classe(Objeto) a ser escrito. Isso evita que informações fluam para classificações mais baixas que a do sujeito.
    - **Exemplo de violação:** Um usuário AS pode fazer copia de um objeto com a classificação AS e depois escrevê-lo de volta como um novo objeto NC, tornando-o visível para todo o sistema.

Segurança e Autorização

- **Controle de inferência (Controles estatísticos de BDs)**
  - Não podem permitir acesso a dados individuais.
  - Devem permitir acesso apenas a consultas que envolvem consultas estatísticas como: COUNT, SUM, MIN, MAX, AVG, etc.
  - Precisa-se ter cuidado, pois através de consultas estatísticas, indiretamente pode-se ter acesso a dados (através do WHERE).
  - Não permitir consulta estatística que o número de tuplas da população especificada pela condição de seleção ficar abaixo de algum limiar.
  - Não permitir sequências de consultas que se refiram repetidamente à mesma população de tuplas.

Segurança e Autorização

- **Ameaça: SQL Injection**
  - Técnica de ataque que utiliza comandos SQL para afetar indevidamente um ambiente computacional.
  - Explora falhas de desenvolvimento em aplicações (principalmente em formulários de autenticação WEB) para “injetar” comandos SQL indevidos no servidor de banco de dados.
  - Como evitar?
    - Controle de qualidade e testes de aplicações
    - Estabelecimento de política de segurança rigorosa (ex: descrever o que é permitido é sempre mais seguro que descrever o que é proibido)

Segurança e Autorização

- **Ameaça: SQL Injection (um exemplo simples)**
  - Imagine uma aplicação web em PHP que, na tela de autenticação solicita usuário e senha, colocando essas informações em duas variáveis (varusuario e varsenha) que são concatenadas em um comando SQL da seguinte forma
    select * from users where username = 'varusuario' and password = 'varsenha';
  - Agora imagine que o invasor informou o seguinte conteúdo na tela para a variável varusuario:

    '; drop table users --
  - Após a concatenação das variáveis o comando SQL ficará da seguinte forma:

    select * from users where username = ''; drop table users --' and password = 'varsenha';

Segurança e Autorização

- **Ameaça: SQL Injection (um exemplo simples - continuação)**
  - Note que o SELECT será executado, mas não retornará nenhuma linha e depois a tabela **users** será destruída. O resto do comando será ignorado pois será transformado em comentário.

    select * from users where username = '; drop table users --' and password = 'varsenha';

  - Com paciência, criatividade e conhecimento da estrutura interna do SGB, um invasor pode causar muito dano ao servidor e, dependendo da segurança do ambiente, até enviar comandos para outros servidores da rede.

Segurança e Autorização

- **Ameaça: SQL Injection (proteção)**
  - Dicas:
    - Validar sempre os dados digitados pelo usuário antes de enviar ao SGBD – rejeitar informação reconhecidamente inválida ou aceitar apenas informação reconhecidamente válida (melhor).
    - Criar usuários com permissões adequadas, evitando usuários genéricos ultrapoderosos e senhas fracas ou óbvias (ex: usuário “admin” e senha “admin”).
    - Nunca permitir que o servidor SQL retorne mensagens diretamente para o usuário, impedindo assim o invasor de extrair informações indevidas;
    - Habilitar logs de segurança no servidor

Leituras Recomendadas

- Introdução a Sistemas de Banco de Dados (Date)
- Sistemas de Banco de Dados (Korth & Silberschatz)
- Sistemas de Banco de Dados Fundamentos e Aplicações (Elmasri & Navathe)
- https://www.devmedia.com.br/sql-injection/6102