Segurança em Banco de Dados

André Luis Schwerz
Rafael Liberato Roberto

Tópicos

- Introdução
- Considerações gerais
- Controle Discricionário
- Controle Mandatário
- Injeção de SQL
- Criptografia

Introdução

- Questões de **segurança** são frequentemente associados a questões de **integridade** de dados.

- Na realidade são bastante diferentes.

- **Segurança:**
  - refere-se à proteção de dados contra revelações, alterações ou destruição não autorizadas.

- **Integridade:**
  - refere-se à exatidão ou validade desses dados.

Introdução

- Segurança:
  - significa proteger os dados contra usuários não autorizados.
- Integridade
  - significa protegê-los contra usuários autorizados

Considerações gerais

- Há vários aspectos a serem considerados no problema de segurança, tais como:
  - Aspectos legais, sociais e éticos
    - Exemplo: a pessoa que faz a solicitação, digamos quanto ao crédito do cliente, tem direito legal à informação solicitada?
  - Controles físicos
    - Exemplo: a sala do computador ou terminal é trancada ou protegida de algum outro modo?
  - Questões de normas
    - Exemplo: como a empresa proprietária do sistema decide quem deve ter acesso a que?
  - Problemas operacionais
    - Exemplo: se é utilizado um esquema de senha, como é conservado o segredo das próprias senhas? Com que frequência elas são alteradas?

Considerações gerais

- Controle de hardware
  - Exemplo: a unidade de processamento fornece recursos de segurança tais o como chaves de proteção de armazenamento ou um modo protegido de operação?

- Suporte do sistema operacional
  - Exemplo: o sistema operacional subjacente apaga o conteúdo da memória principal e dos arquivos de disco quando termina de trabalhar com eles?

- E finalmente questões que são do interesse específico do próprio sistema de banco de dados
  - Exemplo: o sistema de banco de dados tem um conceito de propriedade de dados?

Considerações gerais

- Hoje vamos considerar os aspectos do banco de dados (referente a propriedade dos dados).

- Em geral há duas abordagens gerais de segurança:
  - Controle Discricionário (DAC, do inglês *Discretionary access control*)
  - Controle Mandatário (MAC, do inglês *Mandatory access control*)

Controle discricionário - DAC

- Politica de controle de acesso determinada pelo proprietário (*owner*) do recurso (ex.: uma relação)

- O proprietário decide qual privilégio um usuário tem sobre um determinado recurso.

- Conceitos importantes:
  - Todo objeto em um sistema deve ter um proprietário. A política de acesso é determinada pelo proprietário do recurso. Teoricamente um objeto sem um proprietário é considerado não protegido.
  - Direitos de acesso são estabelecidos pelo proprietário do recurso, que pode inclusive transferir essa propriedade.

- Um exemplo de DAC são as permissões tradicionais do sistema UNIX, implementadas também no Linux.

Controle mandatório - MAC

- Política de acesso determinada pelo sistema e não pelo proprietário do recurso. Este controle é utilizado em sistemas de cujos dados são altamente sensíveis, como governamentais e militares.

- Rótulos de sensibilidade:
  - Todos os usuários e objetos devem ter rótulos associados. Um rótulo de sensibilidade de um sujeito define o seu nível de confiança.
  - Um rótulo de sensibilidade de um objeto define o nível de confiança necessário para acessá-lo.
  - Para acessar um determinado objeto, o sujeito deve ter um rótulo de sensibilidade igual ou superior ao requisitado pelo objeto.

- Sistema baseados em regras (*constraints*)

Controle Discricionário

- A maioria dos SGBDs admite o controle discricionário.
- É necessário para fazer esse controle uma linguagem que admita a definição de restrições de segurança (discricionárias).
- Por razões óbvias, é mais fácil declarar o que é permitido do que enunciar o que não é permitido;
- Por essa razão, as linguagens em geral admitem a definição não de restrições de segurança em si, mas de **autoridades**, que são efetivamente é o oposto das restrições de segurança (se algo é autorizado, não é restringido.)

Controle Discricionário

- Suponha que o DBA cria quatro usuários:
  - A1, A2, A3, e A4

```
CREATE USER A1 WITH PASSWORD 'a1';
```

- Além disso, somente A1 deve ser capaz de criar relações.
- Em SQL2, o DBA deve:

```
CREATE SCHEMA EXAMPLE AUTHORIZATION A1;
```

- O usuário A1 pode criar tabelas sob o esquema chamado **EXAMPLE**

Controle Discricionário

- Suponha que A1 cria duas tabelas **EMPLOYEE** e **DEPARTMENT**
  - A1 é o proprietário (*owner*) dessas duas relações e portanto tem todos os privilégios sob elas.

EMPLOYEE

Name	Ssn	Bdate	Address	Sex	Salary	Dno

DEPARTMENT

Dnumber	Dname	Mgr_ssn

- Suponha que A1 quer conceder A2 o privilégio para inserir e remover tuplas em ambas relações, mas A1 não deseja que A2 possa propagar esses privilégios para outras contas:

```
GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2;
```

Controle Discricionário

- Suponha que A1 quer permitir A3 recuperar informações de suas relações e ser capaz de propagar privilégio SELECT para outros usuários:

```
GRANT SELECT ON EMPLOYEE, DEPARTMENT
    TO A3 WITH GRANT OPTION;
```

- A3 pode conceder o privilégio SELECT na relação EMPLOYEE para A4 ao atribuir:

```
GRANT SELECT ON EMPLOYEE TO A4;
```

- Observe que A4 não pode propagar o privilégio SELECT porque GRANT OPTION não foi dado para ele.

Controle Discricionário

- Suponha que A1 decide anular o privilégio SELECT de A3 na relação EMPLOYEE;

```
REVOKE SELECT ON EMPLOYEE FROM A3;
```

- O SGBD deve agora automaticamente anular o privilégio SELECT de A4, porque A3 não possui mais o privilegio concedido.

Controle Discricionário

- Suponha que A1 que devolver A3 uma capacidade limitada para SELECT na relação EMPLOYEE e quer permitir A3 ser capaz de propagar esse privilégio.
  - A limitação é recuperar somente os atributos NAME, BDATE e ADDRESS e apenas para a tupla com DNO = 5

- A1 então cria uma visão:

```
CREATE VIEW A3EMPLOYEE AS
       SELECT NAME, BDATE, ADDRESS
       FROM EMPLOYEE
       WHERE DNO = 5;
```

- Após a visão criada, A1 pode conceder SELECT da visão A3EMPLOYEE para A3:

```
GRANT SELECT ON A3EMPLOYEE TO A3 WITH GRANT OPTION;
```

Controle Discricionário

- Finalmente, suponha que A1 quer permitir A4 atualizar somente o atributo SALARY de EMPLOYEE;
- A1 pode:

```
GRANT UPDATE ON EMPLOYEE (SALARY) TO A4;
```

  - Os privilégios **UPDATE** e **INSERT** podem descrever atributos específicos que pode ser atualizados ou inseridos em uma relação.
  - Outros privilégios (**SELECT**, **DELETE**) não podem ter atributos específicos.

Resumo

- **GRANT**: passa privilégios de seus objetos para outros usuários

```
GRANT <privilege list>
ON <database objects>
TO <user list>
```

- **REVOKE**: cancela os privilégios de seus objetos a partir de outros usuários

```
REVOKE <privilege list>
ON <database objects>
FROM <user list>
```

Limites na Propagação dos Privilégios

- Técnicas para limitar a propagação de privilégios tem sido desenvolvidas, embora elas ainda não tenham sido implementadas na maioria dos SGBD e não estão no SQL.
  - Limitar a propagação horizontal para um número inteiro *i* significa que um usuário B pode conceder GRANT OPTION para no máximo *i* outros usuários;
  - Propagação vertical é mais complicada; ela limita a profundidade da concessão dos privilégios.

Controle mandatório

- São aplicáveis em banco de dados com uma estrutura de classificação bastante rígida.
  - Aplicações do governo, militares, de inteligência, coorporativas e industriais.
- A ideia básica é que cada objeto possui um nível de classificação.
  - Top secret (TS), secret(S), confidential (C), unclassified (U).
- Cada usuário possui um nível de liberação.
- Níveis podem formar uma ordenação estrita.
  - $TS \geq S \geq C \geq U$

Controle mandatório

- Modelo Bell-LaPadula:
- O usuário *i* só pode **ver** o objeto *j* se o nível de liberação de *i* é maior ou igual ao nível de classificação de *j*.
  - *Classe(i) ≥ Classe(j)*
- O usuário *i* só pode **gravar** o objeto *j* se o nível de liberação de *i* é menor ou igual ao nível de classificação de *j*.
  - *Classe(i) ≤ Classe(j)*

Controle mandatório

- O usuário *i* só pode **gravar** o objeto *j* se o nível de liberação de *i* é menor ou igual ao nível de classificação de *j*.

- Qualquer coisa escrita pelo usuário *i* adquire automaticamente um nível de classificação igual ou superior ao nível de liberação de *i*.

- Essa regra é necessária para evitar, por exemplo, que um usuário com classificação “Top Secret” copie dados secretos em um arquivo com classificação inferior, subvertendo assim o objetivo do esquema de classificação.

Controle mandatório

- Suponha a classificação:
  - TS ≥ S ≥ C ≥ U
- Um usuário com nível de liberação S, pode ver:

(a) EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Brown C	80000 S	Good C	S

- Um usuário com nível de liberação C, pode ver:

(b) EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	NULL C	C
Brown C	NULL C	Good C	C

Controle mandatório

- Suponha a classificação:
  - TS ≥ S ≥ C ≥ U
- Um usuário com nível de liberação U, pode ver:

(c) EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	NULL U	NULL U	U

Controle mandatório

- Suponha que um usuário com nível de liberação C tente atualizar o valor JobPerformance de Smith para Excellent pelo SQL:

```
UPDATE EMPLOYEE
SET JobPerformance = 'Excellent'
WHERE name = 'Smith'
```

- Solução é conhecida como Poli-instanciação.

(d) EMPLOYEE

Name	Salary	JobPerformance	TC
Smith U	40000 C	Fair S	S
Smith U	40000 C	Excellent C	C
Brown C	80000 S	Good C	S

Injeção de SQL

- Conhecido como *SQL Injection*

- É uma ameaça de segurança que se aproveita de falhas em sistemas que interagem com bases de dados via SQL.

- A injeção de SQL ocorre quando o atacante consegue inserir uma série de instruções SQL dentro de uma consulta (query) através da manipulação das entradas de dados de uma aplicação.

Injeção de SQL - Exemplo

- Suponha a sentença:

```
SELECT id, fname, lname
FROM students
WHERE fname = 'Josh' AND lname = 'Smith';
```

- Agora, suponha que o usuário digitou Jo'sh e Smith:

```
SELECT id, fname, lname
FROM students
WHERE fname = 'Jo'sh' AND lname = 'Smith';
```

- Finalmente, veja a injeção de SQL:

```
SELECT id, fname, lname
FROM students
WHERE fname = 'Jo'; DROP TABLE students; --' AND lname = '';
```

Injeção de SQL – mais exemplo

- Suponha a sentença:

```
SELECT *
FROM users
WHERE login = 'Josh' AND pass = '12345';
```

- Veja a Injeção de SQL:

```
SELECT *
FROM users
WHERE login = 'anything' OR '1' = '1' AND pass = '';
```

Injeção de SQL – Técnicas de Proteção

- Utilização de parâmetros:

```
PrepareStatement stmt = conn.prepareStatement("SELECT * FROM
EMPLOYEE WHERE id = ? AND pass = ?");
Stmt.setString(1, id);
Stmt.setString(2, pass);
```

- Remoção de caracteres especiais
  - Por exemplo, o caractere ' deve ser substituído por ".
  - Uso de caracteres de escape.
- Limitação da quantidade de caracteres

Criptografia

- Introdução
  - Até agora dissemos que qualquer candidato a invasor estaria utilizando os recursos normais do sistema para obter acesso ao banco de dados.
  - Agora voltemos nossa atenção para um “usuário” que tenta contornar ilegalmente o sistema.
    - Removendo fisicamente o banco de dados
    - Penetrando em uma linha de comunicação

Criptografia

- Introdução
  - Até agora dissemos que qualquer candidato a invasor estaria utilizando os recursos normais do sistema para obter acesso ao banco de dados.
  - Agora voltemos nossa atenção para um “usuário” que tenta contornar ilegalmente o sistema.
    - Removendo fisicamente o banco de dados
    - Penetrando em uma linha de comunicação
  - A contramedida mais eficaz diante de tais circunstâncias seria a **criptografia de dados**.
  - **Criptografia** é a conversão de dados para um formato, chamado **texto cifrado**, que não pode ser facilmente entendido por pessoas não autorizadas.

Criptografia

- Definições
  - Texto cifrado:
    - Dados criptografados (codificados).
  - Texto limpo
    - Dados inteligíveis que têm significado.
  - Criptografia
    - Processo de transformar texto limpo em texto cifrado.
  - Descriptografia
    - Processo de transformar texto cifrado de volta para texto limpo.

Criptografia

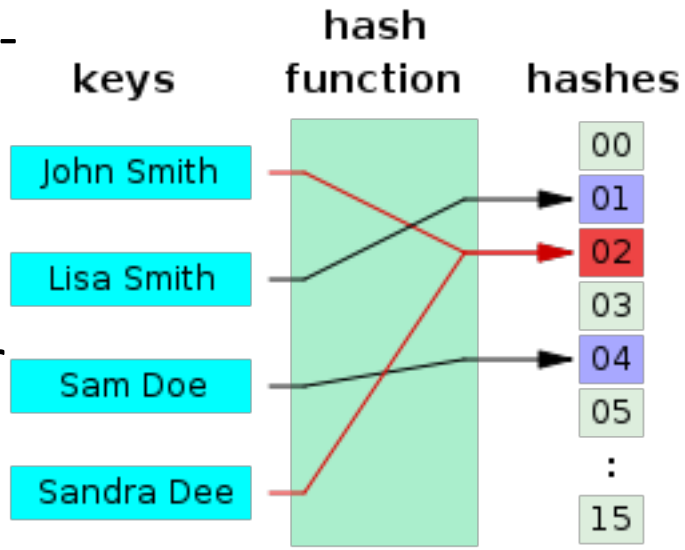
- Confidencialidade da mensagem
  - apenas o destinatário autorizado deve ser capaz de extrair o conteúdo da mensagem da sua forma cifrada.
  - a obtenção de informação sobre o conteúdo da mensagem (como uma distribuição estatística de certos caracteres) não deve ser possível, uma vez que, se o for, torna mais fácil a análise criptográfica.
- Integridade da mensagem
  - o destinatário deverá ser capaz de determinar se a mensagem foi alterada durante a transmissão.
- Autenticação do remetente
  - o destinatário deverá ser capaz de identificar o remetente e verificar que foi mesmo ele quem enviou a mensagem.
- Não-repúdio ou irretratabilidade do emissor
  - não deverá ser possível ao emissor negar a autoria da mensagem.

Criptografia *hash*

- Permite que, através de uma string de qualquer tamanho, seja calculado um identificador digital de tamanho fixo, chamado de valor *hash*.

- O valor *hash* geralmente é formado por 16 bytes (no caso do MD-2, MD-4 e MD-5) ou 20 bytes (no caso do SHA-1).

- Seja uma função *hash H*, e *x* uma string qualquer, teremos que *H(x)* será o valor hash para a string *x*.

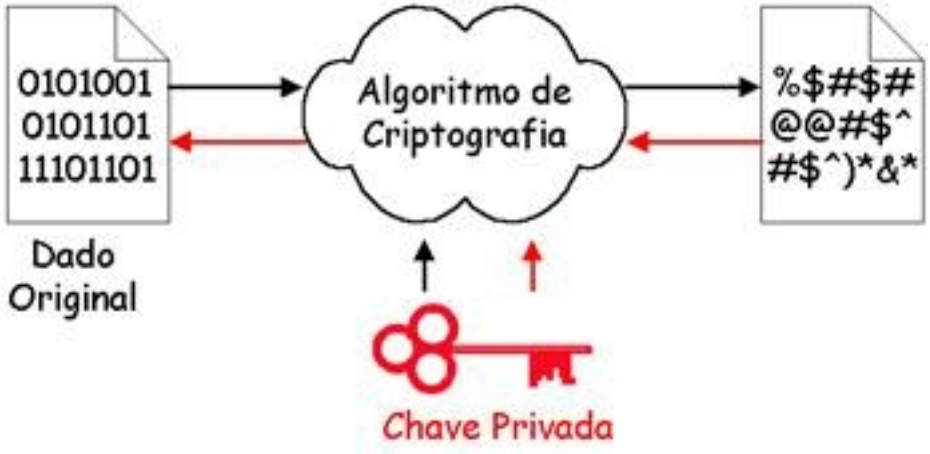


Criptografia *hash*

- As características básicas de uma função *hash* são:
  - O valor de entrada da função possui qualquer tamanho;
  - O valor de saída da função possui tamanho fixo;
  - $H(x)$ é relativamente fácil de ser computado, para qualquer valor de x;
  - $H(x)$ é uma função "*one-way*";
    - uma vez obtido o valor *hash* h para uma string x, é computacionalmente impossível fazer o processo inverso, ou seja, encontrar um valor x tal que $H(x) = h$.
  - $H(x)$ é livre de colisão.
    - as funções *hash* devem garantir uma probabilidade mínima de que duas strings diferentes acabem por resultar no mesmo valor *hash*.
    - Uma alteração na string original que deu origem ao identificador digital, mesmo que de um único bit, acabará por gerar uma alteração significativa no valor *hash* final.

Chave simétrica

- É o tipo mais simples de criptografia
- O emissor e o receptor da mensagem possuem a mesma chave
  - A mesma chave é usada tanto na codificação quanto na decodificação.



- Para ser realizada, basta que o emissor, antes de enviar a mensagem criptografada, envie a chave privada que será utilizada para descriptografá-la.

Chave simétrica

- Vantagem
  - Algoritmos mais rápidos do que os algoritmos assimétricos.

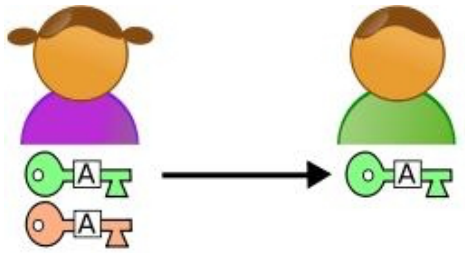
- Desvantagem
  - Necessidade da troca das chaves
  - Impossibilidade de serem usados com fins de autenticação

- *DES (Data Encryption Standard)*
  - Criado pela IBM em 1977,
  - Usa criptografia de 56 bits, o que corresponde a cerca de 72 quadrilhões de chaves diferentes.
  - Foi quebrado por em 1997 por força bruta (tentativa e erro), em um desafio feito na Internet.
- *IDEA (Internacional Data Encryption Algorithm)*
  - Criado em 1991 por Massey e Xuejia Lai,
  - Chaves de 128 bits
  - Possui uma implementação mais simples do que DES
- *RC (Ron's Code ou Rivest Cipher)*
  - Desenvolvido por Ron Rivest,
  - É largamente utilizado em trocas e-mails.
  - Possui diversas versões (RC2, RC4, RC5 e RC6), com chaves que vão de 8 à 1024 bits

Chave assimétrica

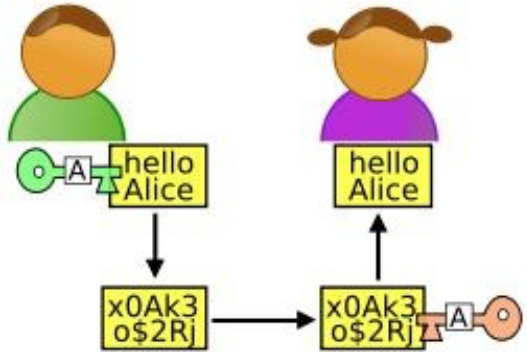
- Há 2 chaves:
  - pública e privada.

- O sistema funciona da forma que alguém cria uma chave e envia essa chave à quem quiser mandar informações à ela, essa é a chamada chave pública.



- Com ela é feita a codificação da mensagem.

- Para decodificação será necessário utilizar uma outra chave que deve ser criada, a chave privada – que é secreta.



Áreas de investigação

- Privacidade e Preservação
- Qualidade dos dados
- Direitos de propriedade intelectual
- Recuperação a ataques