C207 - Banco de Dados I
Fabio de Moura Camargo Neto
github.com/bio353

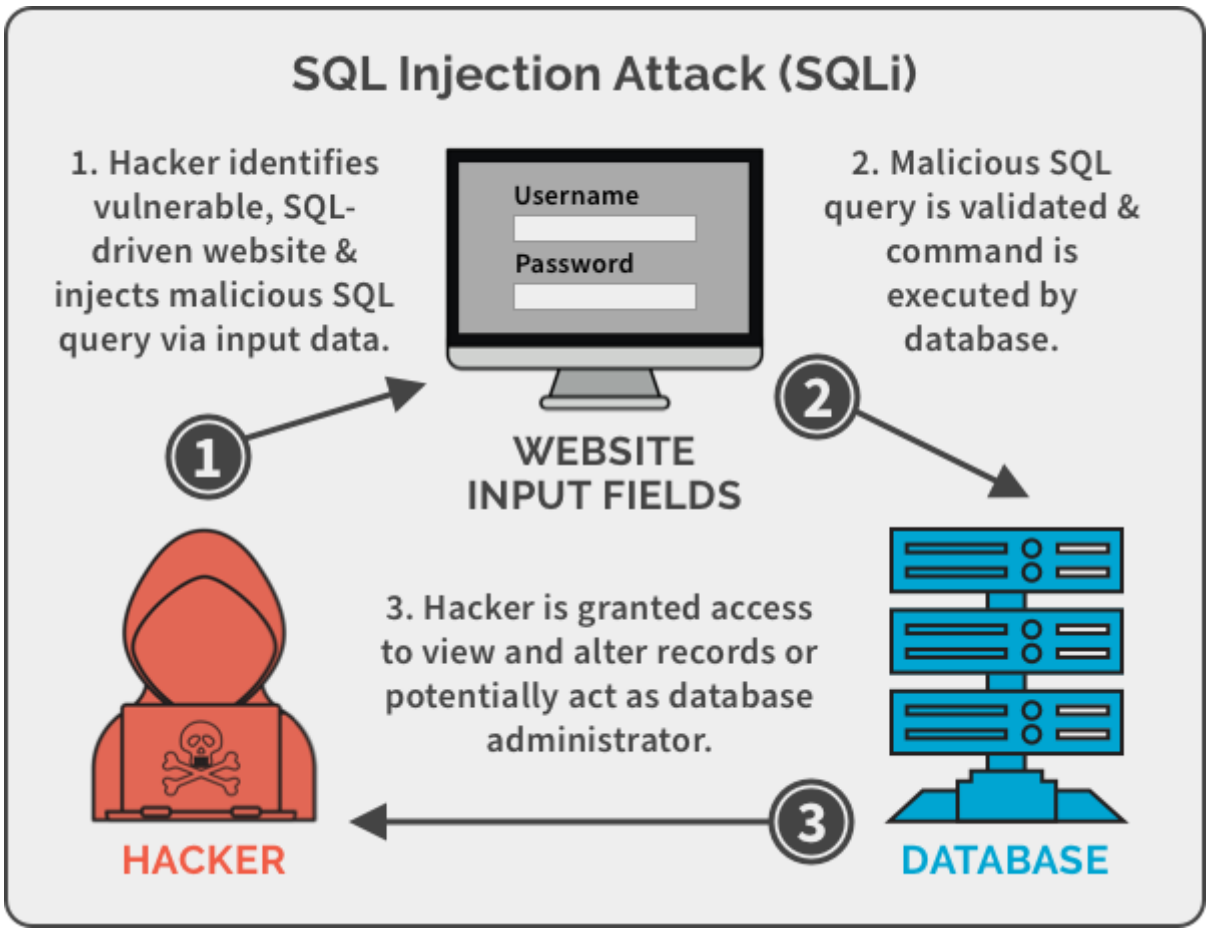
Controle de Acesso

Introdução

Motivo de Controlar o Acesso

Uma vez que um banco de dados é acessado por diversos usuários em diferentes contextos, como administradores de bancos de dados e usuários de aplicações, é importante limitarmos as ações de cada um deles. Não queremos que qualquer um tenha a capacidade de excluir o banco de dados, por exemplo, não é?

Perigo da Injeção de SQL



Comandos `GRANT` e `REVOKE`

Estes comandos servem, respectivamente, para garantir ou revogar privilégios de um usuário. Dentre os privilégios que podemos atribuir, estão:

- `INSERT`, `SELECT`, `UPDATE`, `DELETE`, (CRUD)
- `CREATE`, `ALTER`, `DROP`, `TRIGGER`, `CREATE ROUTINE`, `CREATE VIEWS`
- `ALL` (privilégios “administrativos”)
- `GRANT OPTION` (dar privilégios a outros)
- `USAGE` (acesso apenas para uso)

Para mais privilégios, veja a documentação.

Criando Novos Usuários

Antes de mais nada, precisamos criar um usuário, obviamente. Para criarmos um usuário em um servidor, podemos rodar o seguinte comando:

```
CREATE USER nome_do_usuario@localhost;
```

Já para excluirmos um usuário, basta rodarmos algo como:

```
DROP USER nome_do_usuario@localhost;
```

Por fim, para visualizarmos nossos usuários no **MySQL Workbench**, devemos ir na barra no canto superior esquerdo e clicar em *Server → Users and Privileges*.

Obs.: `localhost` pode ser substituido pelo seu *hostname*, isto é, um nome que define seu banco de dados. Se desejarem mais informações, vocês podem encontrá-las aqui.

Mostrando Permissões

Para vermos quais privilégios um usuário possui, usamos:

```
SHOW GRANTS FOR nome_do_usuario@localhost;
```

Exemplos Práticos

Exemplos de `GRANT`

```
GRANT INSERT, SELECT, UPDATE, DELETE
ON hogwarts.escola
TO usuario1@localhost;

GRANT ALL ON *.*
TO usuario2@localhost;

SELECT(nome, sobrenome, idade), UPDATE(profissao)
ON hogwarts.bruxo
TO usuario3@localhost;
```

Obs.: Para permitirmos determinada ação em todos os bancos de dados e tabelas de um servidor, usamos `*.*`.

Exemplos de `REVOKE`

```
REVOKE INSERT, UPDATE
ON hogwarts.escola
FROM usuario1@localhost;

REVOKE UPDATE
FROM usuario1@localhost;

REVOKE ALL, GRANT OPTION
FROM usuario2@localhost;
```

Exercícios

1. Garanta acesso de um usuário de nome `hermione` apenas para uso em qualquer banco de dados do servidor.
2. Dê todos os privilégios do banco de dados de Hogwarts para o usuário `dumbledore`.
3. Retire o privilégio do usuário `ministro` de criar tabelas no banco de dados `hogwarts`.
4. Permita ao usuário `hagrid` adicionar e alterar os dados da tabela de animais, além de dar privilégios a outros usuários.
5. Dê aos usuários `mcgonagall` e `snape` o poder de alterar o nome e sobrenome dos bruxos e buscar qualquer informação.

6. Retire todos os privilégios do usuário `doloresumbridge`, incluso o poder de dar privilégios a outros usuários.
7. Retire o privilégio do usuário `harrypotter` de adicionar e de excluir participantes do time de quadribol.
8. Mostre os privilégios do usuário `harrypotter`.

```
CREATE DATABASE IF NOT EXISTS hogwarts;
USE hogwarts;
-- Criando Tabela de Escola
CREATE TABLE IF NOT EXISTS Escola (
    id INT AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    localizacao VARCHAR(100) NOT NULL,
    ano_fundacao INT NOT NULL,
    PRIMARY KEY (id)
);
-- Criando Tabela de Casa
CREATE TABLE IF NOT EXISTS Casa (
    id INT AUTO_INCREMENT,
    nome VARCHAR(100) NOT NULL,
    nome_fundador VARCHAR(100) NOT NULL,
    nome_fantasma VARCHAR(100) NOT NULL,
    id_Escola INT NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT fk_Casa_Escola1
        FOREIGN KEY (id_Escola)
        REFERENCES Escola (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
-- Criando Tabela de Animal
CREATE TABLE IF NOT EXISTS Animal (
    id INT AUTO_INCREMENT,
    nome VARCHAR(50) NOT NULL,
    especie VARCHAR(50) NOT NULL,
    PRIMARY KEY (id)
);
-- Criando Tabela de Varinha
CREATE TABLE IF NOT EXISTS Varinha (
    id INT AUTO_INCREMENT,
    tipo_madeira VARCHAR(50) NOT NULL,
    tipo_nucleo VARCHAR(50) NOT NULL,
    tamanho FLOAT NOT NULL,
    PRIMARY KEY (id)
);
-- Criando Tabela de Bruxo
CREATE TABLE IF NOT EXISTS Bruxo (
    id INT AUTO_INCREMENT,
    nome VARCHAR(50) NOT NULL,
    sobrenome VARCHAR(100) NOT NULL,
    profissao VARCHAR(25) NOT NULL,
    idade INT UNSIGNED NOT NULL,
    habilidade_especial VARCHAR(100) NOT NULL,
```

```
    id_Casa INT NOT NULL,
    id_Animal INT DEFAULT NULL,
    id_Varinha INT NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT fk_Bruxo_Casa1
        FOREIGN KEY (id_Casa)
        REFERENCES Casa (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_Bruxo_Animal1
        FOREIGN KEY (id_Animal)
        REFERENCES Animal (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_Bruxo_Varinha1
        FOREIGN KEY (id_Varinha)
        REFERENCES Varinha (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
-- Criando Tabela de Relacionamento (Bruxo - Bruxo)
CREATE TABLE IF NOT EXISTS Bruxo_ensina_Bruxo (
    id_Bruxo_professor INT NOT NULL,
    id_Bruxo_aluno INT NOT NULL,
    PRIMARY KEY (id_Bruxo_aluno, id_Bruxo_professor),
    CONSTRAINT fk_Bruxo_has_Bruxo_Bruxo1
        FOREIGN KEY (id_Bruxo_professor)
        REFERENCES Bruxo (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT fk_Bruxo_has_Bruxo_Bruxo2
        FOREIGN KEY (id_Bruxo_aluno)
        REFERENCES Bruxo (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
-- Crianco Tabela de Time
CREATE TABLE IF NOT EXISTS Time (
    id INT AUTO_INCREMENT,
    nome_apanhador VARCHAR(100) NOT NULL,
    nome_goleiro VARCHAR(100) NOT NULL,
    nome_treinador VARCHAR(100) NOT NULL,
    num_vassouras INT UNSIGNED NOT NULL,
    id_Escola INT NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT fk_Time_Escola1
        FOREIGN KEY (id_Escola)
        REFERENCES Escola (id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```