SQL Injection

Sql injection é um tipo de ataque a sistemas que utilizam banco de dados aproveitando falhas na interface com o usuário para alterar os comandos que serão executados pela aplicação.

SQL Injection

Normalmente as aplicações montam os comando que serão enviados para o SGBD usando o conteúdo de campos editados pelo usuário para completar os comandos.

Por exemplo:

```
sentenca = 'SELECT * FROM cliente WHERE codigo =
'"+cod_cliente+"'"
```

A variável pode ser obtida de um campo editado pelo usuário e deve conter o código do cliente que deve ser pesquisado. Porém, o campos pode ser preenchido com valores que alterem o comando a ser executado como por exemplo:

```
cod_cliente = "'; DELETE FROM cliente WHERE '1'='1"
```

Isso transformaria o comando a ser enviado para o SGBD em:

```
SELECT * FROM cliente WHERE codigo = ''; DELETE FROM
cliente WHERE '1'='1'
```

Essa sentença poderia apagar a tabela mesmo não sendo a intenção da aplicação.

SQL Injection

Outro exemplo seria realizar a autenticação do usuário do sistema com uma sentença como:

```
sentenca = 'SELECT * FROM usuarios WHERE nome =
'"+usuario+"' AND senha='"+senha+"'"
```

Mas essa sentença pode ser alterada com variáveis como:

```
Usuario = "' OR 1=1 –"
```

Para evitar esse tipo de ataque, a aplicação deve evitar a possibilidade da alteração dos comandos a serem executados.

Uma possibilidade e o tratamento dos campos editados pelo usuário para inibir o uso de caracteres para adulterar o comando, impedindo a digitação de caracteres que causem esse problema ou substituindo esses caracteres por sequências que não causem o mesmo problema.

Outra forma de evitar esse tipo de ataque é o uso de Prepared Statments.

Prepared Statments

Prepared Statments são comandos previamente preparados pelo SGBD com a determinação de parâmetros para as posições onde serão substituídos os valores editados pelo usuário.

Na execução do comando, os valores passados como parâmetros serão sempre considerados como parte do comando preparado e não poderão adulterar esse comando.

Usualmente os prepared statements são utilizados em aplicações e as linguagens de programação tem funções/objetos próprios para criação dos prepared statments.

PREPARE

O comando **PREPARE** cria um prepared statment.

```
PREPARE name [ ( datatype [, ...] ) ] AS statement
```

Deve ser atribuído um nome para o prepared statment. Esse nome deve ser único na sessão.

No comando a ser executado, os parâmetros são representados por $1, $2, etc. Os tipos de de cada parâmetro podem ser definidos na criação do prepared statment, mas isso não é obrigatório.

Os prepared statments tem validade apenas durante a sessão onde foram criados e automaticamente deixam de existir no final da sessão.

Exemplo:

```
PREPARE getcliente (text) AS SELECT * FROM cliente WHERE
codigo=$1;
```

EXECUTE

O comando **EXECUTE** executa um prepared statment, substituindo os parâmetros do comando.

```
EXECUTE name [ ( parameter [, ...] ) ]
```

Se os tipos dos parâmetros foram determinados na criação do prepared statment, os valores passados devem ser compatíveis com os tipos dos parâmetros.

Exemplo:

```
EXECUTE getcliente('01');

 codigo |    nome     | sobrenome
--------+-------------+------------
 01     | Jose        | Alves
```

DEALLOCATE

O comando **DEALLOCATE** desaloca um prepared statment.

```
DEALLOCATE [ PREPARE ] { name | ALL }
```

Os prepared statments que não forem explicitamente desalocados, serão automaticamente desalocados no final da sessão.

Exemplo:

```
DEALLOCATE getcliente;
```

Prepared Statments

Se uma aplicação executa várias vezes comandos similares, o uso de um prepared statment pode trazer um ganho de performance por evitar que o comando seja planejado várias vezes. A determinação do plano de execução de um prepared statment é feito durante sua criação. Se o comando tem um planejamento mais difícil e uma execução mais rápida, o uso de prepared statment pode trazer um grande ganho de performance. Se o comando tem um planejamento mais fácil e execução mais lenta, o uso de prepared statment terá menor impacto.

Em algumas circunstâncias, a execução de um prepared statments pode ser mais lenta que a execução do mesmo comando sem a preparação. Como o plano de execução é determinado na criação do prepared statment, o otimizador não pode tirar vantagem das estatísticas dos valores dos parâmetros para determinar um plano de execução mais adequado aos valores dos parâmetros.