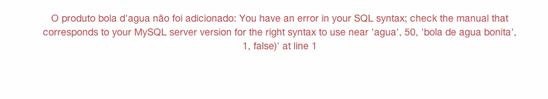
**SQL Injection**

Vamos fazer um teste com um produto novo. Nos logamos e tentamos adicionar uma **Bola D'água**, com uma aspas simples entre **D** e **a**. O que acontece? Dá um erro?



Como assim? Funcionava até agora. Vamos descobrir o que aconteceu? Imprimimos nossa query modificando o**banco-produto**, a função **insereProduto**:

function insereProduto($conexao, $nome, $preco, $descricao, $categoria\_id, $usado) {

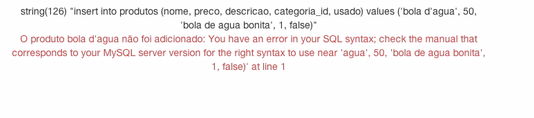
$query = "insert into produtos (nome, preco, descricao, categoria\_id, usado) values ('{$nome}', {$preco}, '{$descricao}', {$categoria\_id}, {$usado})";

var\_dump($query);

return mysqli\_query($conexao, $query);

}

Cadastramos novamente e temos a query:



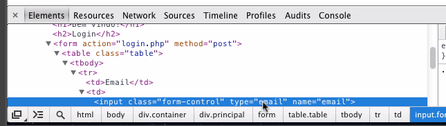
Note que as aspas simples do nome do produto bagunçou a nossa query. O que temos que fazer? Tratar a aspas simples, cuidar de possíveis aspas simples que o usuário pode escrever. Mas não só isso, o usuário pode escrever outros tipos de caracteres ou combinações de caracteres que quebrem nossa query.

Nesse caso vimos que a query quebra mas nada que parece ser uma falha de segurança, mas vamos passar por um segundo exemplo de não tratar as aspas simples (e outros caracteres)?

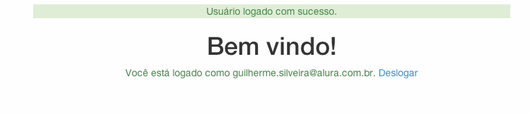
Nos deslogamos e vamos para o formulário de login. Tentamos nos logar com um email malicioso:

' or id=1 or 'guilherme'='

E podemos colocar qualquer senha como **241y79h89fwe98f23nch8hif28jf3**. Tentamos nos logar e ele não deixa. O navegador bloqueou um campo de email inválido pois estamos usando html5. Você pode imaginar que estamos protegidos de um usuário que parece ser malvado. Bom, como usuário final, clico da direita e vou no Developer Tools. Algo bem simples. Clicamos da direita no elemento de campo de texto do email, clicamos da direita, **Inspect Element**.



Agora damos um duplo clique em **type** e mudamos para text. Agora podemos enviar o formulário e, surpresa...



Conseguimos nos logar com o usuário de id 1 mesmo sem saber o email e senha dele. Da mesma maneira que conversamos sobre cookies, não podemos confiar em dados que nosso cliente final envia. O usuário pode inserir um código malicioso para quebrar nossa query de login. Vamos conferir ela? No nosso arquivo **banco-usuario**dentro da função **buscaUsuario** damos um **var\_dump** da query:

<?php

include("conecta.php");

function buscaUsuario($conexao, $email, $senha) {

$senhaMd5 = md5($senha);

$query = "select \* from usuarios where email='{$email}' and senha='{$senhaMd5}'";

var\_dump($query);

die();

$resultado = mysqli\_query($conexao, $query);

$usuario = mysqli\_fetch\_assoc($resultado);

return $usuario;

}

Testamos novamente e o resultado é a query que seleciona o primeiro usuário que tenha o id igual a 1. A precedência dos operadores é utilizada para isolar o **or id=1**.

http://s3.amazonaws.com/caelum-online-public/php2/injection-query.png

Portanto é importantíssimo validarmos a entrada de dados e limparmos ela de quaisquer caracteres que possam quebrá-la. Desejamos evitar que o usuário final possa inserir código sql dentro de nossa query, que ele não possa injetar SQL, que ele não possa fazer **SQL Injection**.

Como fazer isso? Antes de executar a query com o email puro, desejamos tratar as aspas simples etc, desejamos escapar caracteres especiais:

<?php

include("conecta.php");

function buscaUsuario($conexao, $email, $senha) {

$senhaMd5 = md5($senha);

$email = mysqli\_real\_escape\_string($conexao, $email);

$query = "select \* from usuarios where email='{$email}' and senha='{$senhaMd5}'";

var\_dump($query);

die();

$resultado = mysqli\_query($conexao, $query);

$usuario = mysqli\_fetch\_assoc($resultado);

return $usuario;

}

Atualizamos e testamos. A query faz o escape da aspas simples, maravilha. Removemos agora o **var\_dump** e o **die**:

<?php

include("conecta.php");

function buscaUsuario($conexao, $email, $senha) {

$senhaMd5 = md5($senha);

$email = mysqli\_real\_escape\_string($conexao, $email);

$query = "select \* from usuarios where email='{$email}' and senha='{$senhaMd5}'";

$resultado = mysqli\_query($conexao, $query);

$usuario = mysqli\_fetch\_assoc($resultado);

return $usuario;

}

Atualizamos e o código de injection para de funcionar. A mesma coisa temos que fazer sempre que nosso usuário final nos envia dados, jamais confie nos dados do cliente, sempre lembre de cuidar deles.