

SiDi

# SQL INJECTION

Apresentação para entrevista - Mateus de Souza

# Como fazer login sem usuário e senha?

---

E obter acesso a outras informações em um site?

# CONSULTA AO BANCO DE DADOS

Usuário

admin

Senha

admin

Login

```
SELECT id FROM users WHERE password = 'admin' AND username = 'admin'
```

id	username	password
1	admin	admin
2	alice	1234
3	bob	password

# INJEÇÃO NO BANCO DE DADOS

Usuário

**não importa**

Senha

**admin' OR 1=1 --**

Login

id	username	password
1	admin	admin
2	alice	1234
3	bob	password

```
SELECT id FROM users WHERE password = 'admin' OR 1=1 -- ' AND username = 'não importa'
```

```
SELECT id FROM users WHERE password = 'admin' OR 1=1 -- ' AND username = 'não importa'
```

# Exemplo

---

Aplicando SQL Injection:  
Banco AltoroMutual

# IMPACTOS DE UM ATAQUE

---



Exposição de  
dados sensíveis



Acesso ao  
atacante



Comprometimento  
dos dados



Reputação  
manchada

# TIPOS DE ATAQUES

## SUBVERTENDO A LÓGICA

Exemplo apresentado -  
AltoroMutual

Acrescentando ou mudando  
condicionais e retirando outras  
através de comentários

## OBTENDO DADOS DE OUTRAS TABELAS

Exemplo desafio picoCTF -  
[More SQLi](#)

Quando a aplicação retorna os  
resultados de uma consulta, é  
possível alterá-la para retornar  
resultados de outras tabelas

## ATAQUES "CEGOS"

Exemplo lab hackinghub -  
[Boolean Based SQL Injection](#)

Quando a aplicação não retorna  
resultados ou aponta erros, mas  
permite inferir resultados  
dependendo da resposta ou do  
tempo envolvido

# COMO PREVENIR?

01.

## Limitar privilégios

---

Permitir somente o mínimo necessário para contas da aplicação operarem no banco de dados.

Não permitir que o usuário root conecte à aplicação e crie outros usuários.

02.

## Prepared Statements

---

Alterar o código SQL para usar “templates” que são compilados sem as entradas de usuários, garantindo que novas entradas não serão lidas como código.



# COMO PREVENIR?

## 02.

### Prepared Statements

---

Alterar o código SQL para usar “templates” que são compilados sem as entradas de usuários, garantindo que novas entradas não serão lidas como código.

#### Código anterior:

```
SELECT id FROM users WHERE password = 'admin' AND username = 'admin';
```

#### Com prepared statements:

```
PREPARE stmt FROM 'SELECT id FROM users WHERE password = ? AND username = ?';  
SET @password = 'your_password';  
SET @username = 'your_username';  
EXECUTE stmt USING @password, @username;
```

# COMO PREVENIR?

01.

## Limitar privilégios

---

Permitir somente o mínimo necessário para contas da aplicação operarem no banco de dados.

Não permitir que o usuário root conecte à aplicação e crie outros usuários.

02.

## Prepared Statements

---

Alterar o código SQL para usar “templates” que são compilados sem as entradas de usuários, garantindo que novas entradas não serão lidas como código.

03.

## Whitelisting/Typecasting

---

Definir quais valores podem ser inseridos em um campo e compará-los com a entrada do usuário.

Converter a entrada do usuário para o tipo esperado (inteiro, booleano, string) para garantir que não há um valor inválido.

04.

## SQL Sanitization

---

“Limpar” a entrada, removendo caracteres especiais que podem indicar uma injeção, como: ‘ ou “.

Uso de bibliotecas de linguagens de programação com funções de limpeza a serem usadas no código.

# MUITO OBRIGADO!

---

<https://github.com/MateusAMSouza/sql-injection>



[Mateus Alexandre Martins de Souza](#)



[MateusAMSouza](#)



[mateusalexandrem@gmail.com](mailto:mateusalexandrem@gmail.com)