

REPÚBLICA FEDERATIVA DO BRASIL
UNIVERSIDADE HACKER
INTRODUÇÃO A SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS
MÓDULO VII – DESENVOLVIMENTO WEB

SQL Injection

Prof. Alex Dias Camargo

alexcamargoweb@gmail.com



UNIHACKER.CLUB
PROGRAMA UNIVERSIDADE HACKER



I. Plano de aula

Na aula anterior foi visto:

- **Arquitetura *Web***
- **Riscos de uma aplicação**



I. Plano de aula

Nesta aula será apresentado:

- **Conceitos de injeção de dados**
- **Exemplo prático: PHP e *MySQL***



1. Introdução

Os **ataques de injeção** acontecem quando dados não confiáveis são enviados para um interpretador por meio de uma **entrada de formulário** ou **algum outro envio de dados**.

- ❑ **Cenário:** um sistema acadêmico utiliza dados não validados na construção de uma chamada SQL para **exibir o histórico de um aluno**.
- ❑ **Ataque:** Injeção de SQL via GET na URL.
- ❑ **Ambiente:** *Apache 2.4, PHP 5.6, MySQL 5.7.*
- ❑ **Repositório:** <https://github.com/alexcamargoweb/unihacker>





1. Introdução

matricula	nome	curso	media
101010	Fulano de Tal Souza	Sistemas de Informação	8.7
101011	Fulana de Tal Batista	Arquitetura	9.5
111111	Ciclano da Silva Nunes	Sistemas de Informação	6.1
111000	Beltrana Costa e Silva	Ciências Biológicas	9.1
111000	Beltrana Costa e Silva	Bioquímica	8.6

Figura. Banco de dados: visão geral.



1. Introdução

Universidade HaCker - Dados do aluno

Matrícula	Nome	Curso	Média
101010	Fulano de Tal Souza	Sistemas de Informação	8.7

Parâmetro via GET

Figura. Consulta ao banco de dados: acesso padrão.



1. Introdução

← → ↻ 🏠 | 127.0.0.1/unihacker/curso/sql_injection.php?matricula=101010 ' OR 1=1 --'

Universidade HaCker - Dados do aluno

Matrícula	Nome	Curso	Média
101010	Fulano de Tal Souza	Sistemas de Informação	8.7
101011	Fulana de Tal Batista	Arquitetura	9.5
111111	Ciclano da Silva Nunes	Sistemas de Informação	6.1
111000	Beltrana Costa e Silva	Ciências Biológicas	9.1
111000	Beltrana Costa e Silva	Bioquímica	8.6

Parâmetro via GET
+
' OR 1=1 - '

Figura. Consulta ao banco de dados: acesso malicioso.



1. Introdução

```
1  <?php
2
3  // Abre conexão com o banco de dados
4  $conn = mysqli_connect('127.0.0.1', 'unihacker', 'UnIh@CkEr', 'unihacker');
5  // Define o charset
6  mysqli_set_charset($conn, "utf8");
7
8  // Faz a consulta ao banco (COM VULNERABILIDADE) ' OR 1=1 --'
9  $query = "SELECT * FROM alunos WHERE matricula = '".$_GET['matricula']."'";
10 // Executa a query
11 $result = mysqli_query($conn, $query);
12
13 ?>
```

Parâmetro
via GET
sem filtro!

Figura. Consulta ao banco de dados: código com vulnerabilidade.



1. Introdução

```
1 <?php
2
3 // Abre conexão com o banco de dados
4 $conn = mysqli_connect('127.0.0.1', 'unihacker', 'UnIh@CkEr', 'unihacker');
5 // Define o charset
6 mysqli_set_charset($conn, "utf8");
7
8 // Faz a consulta ao banco (SEM VULNERABILIDADE) ' OR 1=1 --'
9 /query = "SELECT * FROM alunos WHERE matricula = '".mysqli_real_escape_string($conn, $_GET['matricula']).'";
10
11 // Executa a query
12 $result = mysqli_query($conn, $query);
13
14 ?>
```

Parâmetro via GET com filtro!

Figura. Consulta ao banco de dados: código com vulnerabilidade filtrada.



1.1

```
28 <html>
29 <head>
30 <title>Universidade Hacker</title>
31 </head>
32 <body style="font-family: Tahoma; font-size:12px;">
33 <div style="border: 3px solid #555; width: 600px; padding:15px;">
34 <span style='font-weight:bold'>Universidade HaCker - Dados do aluno</span>
35 <br /><br />
36 <table>
37 <tr>
38 <td class="dados negrito">Matrícula</td>
39 <td class="dados negrito">Nome</td>
40 <td class="dados negrito">Curso</td>
41 <td class="dados negrito">Média</td>
42 </tr>
43 <?php
44 while($row = mysqli_fetch_assoc($result)){
45 >
46 <tr>
47 <td class="dados"><?php print($row['matricula']); ?></td>
48 <td class="dados"><?php print($row['nome']); ?></td>
49 <td class="dados"><?php print($row['curso']); ?></td>
50 <td class="dados"><?php print($row['media']); ?></td>
51 </tr>
52 <?php
53 }
54 >
55 </table>
56 </body>
57 </html>
```

Figura. Consulta ao banco de dados: restante do código-fonte.



2. Exercícios

1. Responda o *quiz* sobre *SQL Injection*:

SQL Server Questions and Answers - SQL Injection

<https://moodle.unihacker.club/mod/url/view.php?id=173>



Referências básicas

OWASP, Top. Top 10-2017. **The Ten Most Critical Web Application Security Risks. OWASP™ Foundation. The free and open software security community.** URL: https://www.owasp.org/index.php/Top_10-2017_Top_10, 2017.