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





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



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.
- \square Ambiente: Apache 2.4, PHP 5.6, MySQL 5.7.
- Repositório: https://github.com/alexcamargoweb/unihacker





| 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.



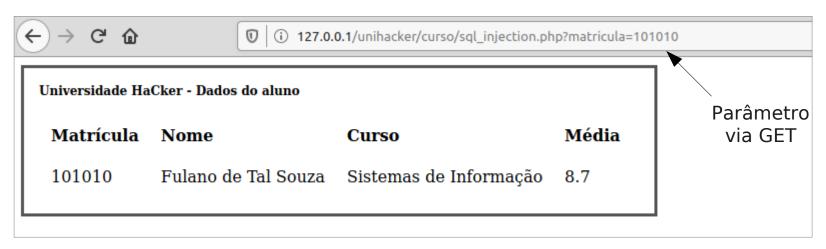


Figura. Consulta ao banco de dados: acesso padrão.



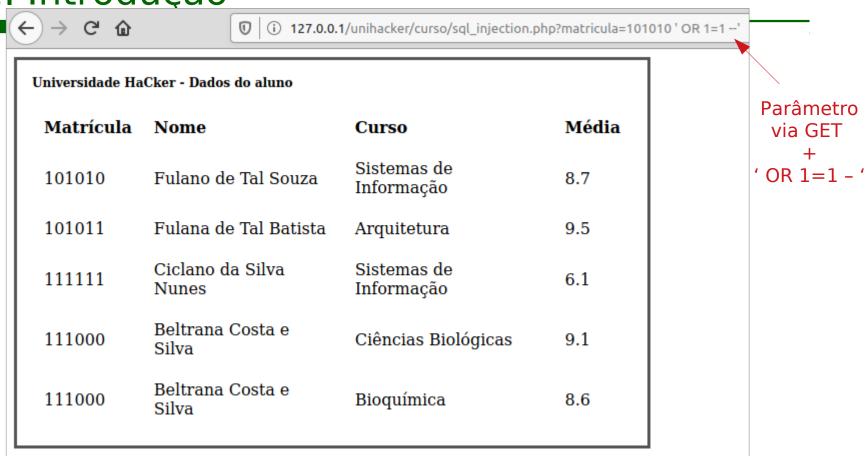


Figura. Consulta ao banco de dados: acesso malicioso.



Parâmetro via GET sem filtro!

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



```
<?php
 2
      // Abre conexão com o banco de dados
     $conn = mysqli connect('127.0.0.1', 'unihacker', 'UnIh@CkEr', 'unihacker');
      // Define o charset
      mysqli set charset($conn, "utf8");
 7
 8
      // Faz a consulta ao banco (SEM VULNERABILIDADE) ' OR 1=1 --'
      /query = "SELECT * FROM alunos WHERE matricula = '".mysqli real escape string($conn, $ GET['matricula'])."'";
 9
10
11
     // Executa a query
                                                                                       Parâmetro
via GET
      $result = mysqli query($conn, $query);
12
13
                                                                                       com filtro!
14
      ?>
```

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



1. |

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