

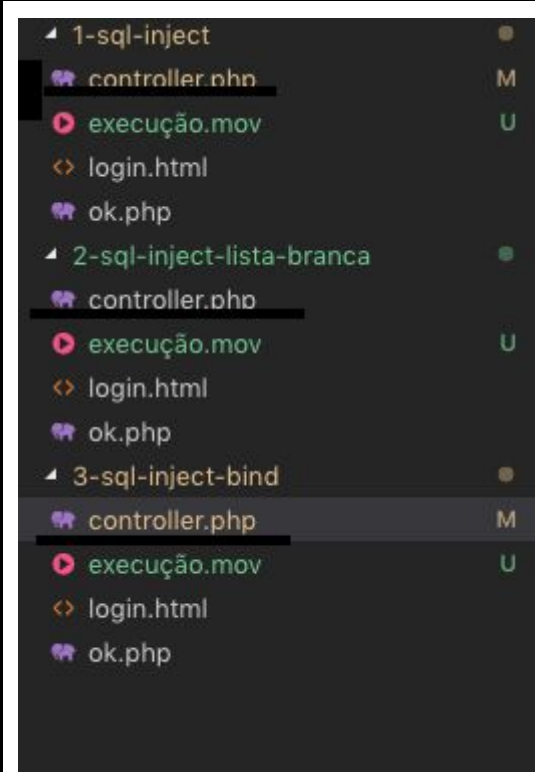
Nome: Jorge Guilherme Kohn, Felipe Anselmo.

A - fragilidade pode ser vista abaixo.

<https://youtu.be/6l2V5txyAJ8>

Questões B - C - Foi utilizado PHP para explorar as vulnerabilidade de SQL-inject, basta extrair as pastas onde o PHP está observando (pasta onde é possível executar scripts PHP).

É preciso apontar a conexão do mysql no PHP, dentro de cada pasta, existe um arquivo controller.php, abra-o e altere a conexão.

	<pre>try { \$conn = new PDO('mysql:host=127.0.0.1;dbname=trab', "root", "root"); \$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); } catch(PDOException \$e) { echo 'ERROR: ' . \$e->getMessage(); }</pre>
--	---

Adicionar a tabela

```
create table if not exists usuario (
    id int auto_increment,
    email varchar(255) not null,
    senha varchar(255) not null,
    primary key (id)
);`
```

```
insert into usuario (email, senha) values('admin@admin', '123')
```

O arquivo de conte a vulnerabilidade é 1-sql-inject;

O arquivo com a lista branca é 2-sql-inject-lista-branca

O arquivo com bind é o 3-sql-inject-bind

D - Fragilidade corrigida

<https://youtu.be/ean7aQ2ygDw>

E - Foi corrigido através de uma lista branca, aceitando os caracteres abaixo.

```
'@', '.', '_', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l',  
'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'x', 'w', 'y', 'z', "1",  
"2", "3", "4", "5", "6", "7", "8", "9", "0"]
```

É transformado em uma array de char os inputs de email e senha, percorrido e somente copiado para uma nova variável os que estão dentro da lista branca.

Ao final comparando se os valores iniciais são iguais aos finais, se for igual efetua a query.

F - Foi utilizado a parametrização oferecida pelo PDO no PHP. Realizando o bind do ':email' e ':senha', com os comandos bindParams.