

Para montar um laboratório de cybersecurity no Linux Debian com um banco de dados MySQL e um formulário PHP que seja vulnerável a ataques como SQL injection e força bruta, você pode seguir os passos abaixo:

1. 1. Instalação e Configuração do MySQL

1. Instale o MySQL Server:

```
sudo apt update  
sudo apt install mariadb-server
```

```
sudo apt update  
sudo apt install mariadb-server
```

2. Configure o MySQL para sua aplicação:

mysql

```
root@debian:~# mysql  
Welcome to the MariaDB monitor.  Commands end with ; or \g.  
Your MariaDB connection id is 30  
Server version: 10.5.26-MariaDB-0+deb11u2 Debian 11  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]>
```

Dentro do MySQL, execute:

```
CREATE DATABASE bd_aluno;  
CREATE USER 'aluno'@'localhost' IDENTIFIED BY 'fiap';  
GRANT ALL PRIVILEGES ON bd_aluno.* TO 'aluno'@'localhost';  
FLUSH PRIVILEGES;  
USE bd_aluno;  
CREATE TABLE tab_aluno (login VARCHAR(255), senha VARCHAR(255));
```

```
MariaDB [(none)]> CREATE DATABASE bd_aluno;  
Query OK, 1 row affected (0,000 sec)  
  
MariaDB [(none)]> CREATE USER 'aluno'@'localhost' IDENTIFIED BY 'fiap';  
Query OK, 0 rows affected (0,012 sec)  
  
MariaDB [(none)]> GRANT ALL PRIVILEGES ON bd_aluno.* TO 'aluno'@'localhost';  
Query OK, 0 rows affected (0,012 sec)  
  
MariaDB [(none)]> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0,001 sec)  
  
MariaDB [(none)]> USE bd_aluno;  
Database changed  
MariaDB [bd_aluno]> CREATE TABLE tab_aluno(login VARCHAR(255), senha VARCHAR(255));  
Query OK, 0 rows affected (0,022 sec)
```

OBS: caso saia do banco, o usuário do banco de dados é aluno e sua senha: fiap.

```
root@debian:~# mysql -u aluno -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 32
Server version: 10.5.26-MariaDB-0+deb11u2 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> _
```

Ver a estrutura da tabela: Para ver a estrutura da tabela `tab_aluno`, use:

`DESCRIBE tab_aluno;`

```
MariaDB [bd_aluno]> describe tab_aluno;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| login | varchar(255)  | YES  |     | NULL    |       |
| senha | varchar(255)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,001 sec)

MariaDB [bd_aluno]>
```

OU `SHOW COLUMNS FROM tab_aluno;`

```
MariaDB [bd_aluno]> show columns from tab_aluno;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| login | varchar(255)  | YES  |     | NULL    |       |
| senha | varchar(255)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0,002 sec)
```

3. Inserindo Dados na Tabela

Primeiro, você precisa inserir alguns registros na tabela para ter dados com os quais trabalhar. Aqui estão os comandos SQL que você pode usar para isso:

`USE bd_aluno;`

`INSERT INTO tab_aluno (login, senha) VALUES ('usuario1', 'senha123');`

`INSERT INTO tab_aluno (login, senha) VALUES ('usuario2', '123456');`

`INSERT INTO tab_aluno (login, senha) VALUES ('usuario3', 'abcdef');`

```

MariaDB [bd_aluno]> INSERT INTO tab_aluno(login, senha) VALUES ('pedro', 'senha123');
Query OK, 1 row affected (0,012 sec)

MariaDB [bd_aluno]> INSERT INTO tab_aluno(login, senha) VALUES ('aluno', 'bowwow');
Query OK, 1 row affected (0,012 sec)

MariaDB [bd_aluno]> INSERT INTO tab_aluno(login, senha) VALUES ('fiap', 'aluno123');
Query OK, 1 row affected (0,012 sec)

MariaDB [bd_aluno]> INSERT INTO tab_aluno(login, senha) VALUES ('fiap', 'aluno');
Query OK, 1 row affected (0,015 sec)

MariaDB [bd_aluno]> INSERT INTO tab_aluno(login, senha) VALUES ('aluno', 'fiap');
Query OK, 1 row affected (0,012 sec)

MariaDB [bd_aluno]>

```

Consultar dados da tabela: Para exibir todos os dados da tabela `tab_aluno`, use o comando:

```
SELECT * FROM tab_aluno;
```

```

MariaDB [bd_aluno]> select * from tab_aluno;
+-----+-----+
| login | senha |
+-----+-----+
| pedro | senha123 |
| aluno | bowwow |
| fiap | aluno123 |
| fiap | aluno |
| aluno | fiap |
+-----+-----+
5 rows in set (0,000 sec)

MariaDB [bd_aluno]> _

```

Configuração do PHP e Apache

1. Instale o Apache e o PHP:

```
root@debian:~# apt install apache2 php libapache2-mod-php php-mysql
Lendo listas de pacotes... Pronto
Construindo árvore de dependências... Pronto
Lendo informação de estado... Pronto
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils php7.4-mysql
Pacotes sugeridos:
```

2. Configure o PHP para usar o banco de dados

Para criar um formulário de login em PHP que seja vulnerável a SQL injection, você precisará de um script que interaja com o banco de dados sem utilizar práticas de sanitização ou preparação de consultas. Aqui está um exemplo básico de como você pode fazer isso:

1. Banco de dados

Primeiro, certifique-se de que seu banco de dados e tabela estão configurados.

BANCO: bd_aluno

login: aluno

senha: fiap

2. Script de Conexão PHP

db.php

```
<?php
$servername = "localhost";
$username = "aluno";
$password = "fiap";
$dbname = "bd_aluno";

// Criando a conexão
$conn = new mysqli($servername, $username, $password, $dbname);

// Checando a conexão
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

```

<?php
$servername = "localhost";
$username = "aluno";
$password = "fiap";
$dbname = "bd_aluno";

// Criando a conexão
$conn = new mysqli($servername, $username, $password, $dbname);

// Checando a conexão
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>

```

3. Formulário HTML

INDEX.HTML

```

<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <form action="login.php" method="post">
        Login: <input type="text" name="login" required><br>
        Senha: <input type="password" name="senha" required><br>
        <input type="submit" value="Entrar">
    </form>
</body>
</html>

```

```

<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <form action="login.php" method="post">
        Login: <input type="text" name="login" required><br>
        Senha: <input type="password" name="senha" required><br>
        <input type="submit" value="Entrar">
    </form>
</body>
</html>

```

4. Script de Login (login.php)

```
<?php
include 'db.php'; // Inclui o script de conexão

$login = $_POST['login'];
$senha = $_POST['senha'];

$sql = "SELECT * FROM tab_aluno WHERE login='$login' AND senha='$senha'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "Login bem sucedido!";
} else {
    echo "Usuário ou senha incorretos!";
}
$conn->close();
?>
```

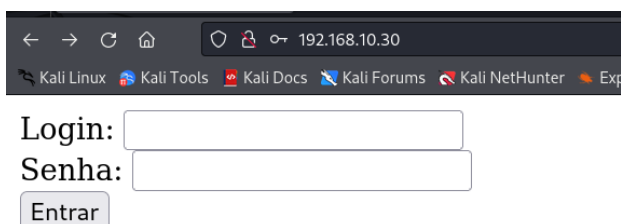
```
<?php
include 'db.php'; // Inclui o script de conexão

$login = $_POST['login'];
$senha = $_POST['senha'];

$sql = "SELECT * FROM tab_aluno WHERE login='$login' AND senha='$senha'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "Login bem sucedido!";
} else {
    echo "Usuário ou senha incorretos!";
}
$conn->close();
?>
```

RESULTADO ESPERADO:

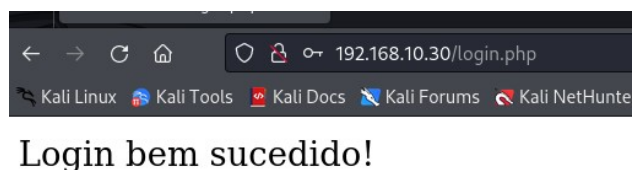


192.168.10.30

Login:

Senha:

Entrar



192.168.10.30/login.php

Login bem sucedido!

Agora iremos realizar o ataque de força bruta:

VAMOS AGORA ACESSAR O SITE.

https://github.com/payloadbox/sql-injection-payload-list/blob/master/Intruder/exploit/Auth_Bypass.txt