SQLインジェクション Ver 2.1

2025 年 10 月 23 日

1 準備作業

1.1 動作させるファイル

次の 2 つのファイルを用意する。

index.php

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <p>idで検索</p>
  <form action="all.php" method="post">
    ID:<input type="text" name="id"><br>
    <input type="submit" value="送信">
  </form>
</body>
</html>
```

```
<?php
function h($str){
  return htmlspecialchars($str, ENT_QUOTES, 'UTF-8');
}

$user = "testuser";
$password = "testuser";
$dbName = "testdb";
$host = "localhost:3306";
$dsn = "mysql:host={$host};dbname={$dbName};charset=utf8";
?>
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <link rel="stylesheet" href="../css/style.css">
</head>
<body>
  <div>
    <?php
    $id = $_POST['id'] ?? '';
    try {
      $pdo = new PDO($dsn, $user, $password);
      $pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
      $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
      echo "データベース{$dbName}に接続しました";
```

```
28
29        $sql = "SELECT * FROM member WHERE id = '{$id}'";
30        $stm = $pdo->prepare($sql);
31        // $stm->bindValue(':id', $id, PDO::PARAM_INT);
32        $stm->execute();
33        $result = $stm->fetchAll(PDO::FETCH_ASSOC);
34      ?>
35        <table>
36          <thead>
37            <tr>
38              <th>ID</th>
39              <th>名前</th>
40              <th>年齢</th>
41              <th>性別</th>
42            </tr>
43          </thead>
44          <tbody>
45            <?php
46            foreach ($result as $row) {
47              echo "<tr>";
48              echo "<td>", h($row['id']), "</td>";
49              echo "<td>", h($row['name']), "</td>";
50              echo "<td>", h($row['age']), "</td>";
51              echo "<td>", h($row['sex']), "</td>";
52              echo "</tr>";
53            }
54            ?>
55          </tbody>
56        </table>
57      <?php
58      } catch (Exception $e) {
59        echo '<span class="error">エラーがありました</span><br>';
60        echo $e->getMessage();
61        exit();
62      }
63      ?>
64      <hr>
65      <a href="index.php">もどる</a>
66    </div>
67  </body>
68  </html>
```

2 未対策のプログラムコードを実行する

2.1 実行の手順

all.php の次の部分を確認する。

```
1  $sql = "SELECT * FROM member WHERE id = '{$id}'";
2  $stm = $pdo->prepare($sql);
3  // $stm->bindValue(':id', $id, PDO::PARAM_INT);
4  $stm->execute();
```

SQL 文の中に $id と変数を埋めこんでいる。

また、$stm->bindValue(...) はコメントにしてある。

2.2 MySQL のログを有効にする

MySQL はどのような SQL 文を実行したのかは、ログをとるとわかる。

MySQL でクエリログを有効にする手順

MySQL ではクエリログは次の手順で有効にできる。
MariaDB [(none)]> SET GLOBAL general_log_file = 'C:/xampp/mysql/log/mysql.log';
MariaDB [(none)]> SET GLOBAL general_log = 1;
この場合、"C:¥xampp¥mysql" に log フォルダを作成しておく必要がある。mysql.log ファイルは自動で作成される。
MySQL を再起動すると、もとにもどる。

ログを有効にしておいて、危険なデータを送ってみる。

```
' or 1 = 1; --
```

なんと、全員が表示された。
MySQL のクエリログを見てみると、次のような SQL が動作したのだとわかる。

```
' or 1 = 1; --
```

この SQL 文を見ると、WHERE 句の id = '' が動作したのではなくて、OR 句の 1 = 1 が動作し、それが条件としてデータを表示したのだということがわかる。

2.3 なぜそうなったのか?

その原因は、プログラムコードに問題がある。

```
$sql = "SELECT * FROM members WHERE id = '{$id}'";
```

文字列の中に 変数$id を埋めこんだのがいけないのである。そのことによって、悪意のあるユーザーに SQL コマンドの実行を許してしまったのである。

3 対策済みのプログラムコードを実行する

3.1 実行の手順

問題の箇所を次のように変える。

```
1 $sql = "SELECT * FROM member WHERE id = :id";
2 $stm = $pdo->prepare($sql);
3 $stm->bindValue(':id', $id, PDO::PARAM_INT);
4 $stm->execute();
```

そして、さっきと同じようにする。

`' or 1 = 1; --` と入力する。
今度は何も表示されない。
ログを見ると、今回は次の SQL が動作したことがわかる。

```
SELECT * FROM members WHERE id = '\' or 1 = 1; -- '
```

"\'" というのは、'(シングルクォーテーション) をエスケープしている。つまり、"' or 1 = 1; -- " が 1 つの文字列として扱われ、その文字列に合致するものを調べるという SQL が動作したのである。

4 プリペアードステートメントの重要性

送信されてきたデータを変数に格納し、それを使って SQL 文を発行するときは、SQL 文の中に変数を埋め込むと、"SQL インジェクション" という攻撃を受ける危険性がある。これは、悪意のあるユーザーに任意を SQL 文の実行を許してしまう。

その対策として、変数を使うときは、SQL 文にプレースホルダーを記述し、変数値をデータとして別個に与えるようにする。そのことにより、ユーザーの入力が SQL 構文として解釈されることがない。

PDO::PARAM_INT って何?

$stm->bindValue(':id', $id, PDO::PARAM_INT) という文で、
PDO::PARAM_INT というのがあるが、
$pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, false); の場合、
PDO::PARAM_INT は動作していない。
与えられたデータが文字列なら文字列のまま、 数値なら数値のまま、MySQL に渡す。
$pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES, true); の場合は、
PDO::PARAM_INT は動作する。
与えられたデータを整数に変換して MySQL に渡す。小数点値なら、小数点以下を切り捨てて MySQL に渡す。